Improving Classification Performance of Microarray Analysis by Feature Selection and Feature Extraction Methods

by

Jing Sun

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science (MSc) in Computational Sciences

The Faculty of Graduate Studies

Laurentian University

Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Faculty of Graduate Studies/Faculté des études supérieures

Title of Thesis Titre de la thèse	Improving Classification Performance of Microarray Analysis by Feature Selection and Feature Extraction Methods
Name of Candidate Nom du candidat	Sun, Jing
Degree Diplôme	Master of Science
Department/Program Département/Programme	Computer Science
Date of Defence Date de la soutenance	octobre 26, 2016

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Kalpdrum Passi
(Supervisor/Directeur(trice) de thèse)

Dr. Ratvinder Grewal
(Committee member/Membre du comité)

Dr. Julia Johnson
(Committee member/Membre du comité)

Dr. Gulshan Wadhwa
(External Examiner/Examinateur externe)

Approved for the Faculty of Graduate Studies
Approuvé pour la Faculté des études supérieures
Dr. Shelley Watson
Madame Shelley Watson
Acting Dean, Faculty of Graduate Studies
Doyenne intérimaire, Faculté des études supérieures

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

I, **Jing Sun**, hereby grant to Laurentian University and/or its agents the non-exclusive license to archive and make accessible my thesis, dissertation, or project report in whole or in part in all forms of media, now or for the duration of my copyright ownership. I retain all other ownership rights to the copyright of the thesis, dissertation or project report. I also reserve the right to use in future works (such as articles or books) all or part of this thesis, dissertation, or project report. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that this copy is being made available in this form by the authority of the copyright owner solely for the purpose of private study and research and may not be copied or reproduced except as permitted by the copyright laws without written authority from the copyright owner.

Abstract

In this study, we compared two feature extraction methods (PCA, PLS) and seven feature selection methods (mRMR and its variations, MaxRel, QPFS) on four different classifiers (SVM, RF, KNN, NN). We use ratio comparison validation for PCA method and 10-folds cross validation method for both the feature extraction and feature selection methods. We use Leukemia data set and Colon data set to apply the combinations and measured accuracy as well as area under ROC. The results illustrated that feature selection and extraction methods can both somehow improve the performance of classification tasks on microarray data sets. Some combinations of classifier and feature preprocessing method can greatly improve the accuracy as well as the AUC value are given in this study.

Keywords

Microarray datasets, Feature Extraction, Feature Selection, Principal Component Analysis, Partial Least Square, minimum Redundancy- Maximum Relevant, Quadratic Programming Feature Selection, Support Vector Machine, Random Forest, k-Nearest-Neighbor, Neural Network.

Acknowledgments

First of all, I would like to give my fully thanks to my supervisor, Dr. Kalpdrum Passi, for his patient and insightful guidance in research, also for his consistent support and encouragement through all stages of my master study. Although there is a lost when I facing my direction, but still he is there and point out a way for me. Guide my step by step, that gave me the definition of a supervisor. Without him, I would not have been able to complete this work.

Secondly, I want to say thanks for my family and my aunt. Without them, I should have been a high school teacher in my hometown. They gave me a way to have an experience of study abroad and live in Canada. They gave me full support in financial and mental.

Table of Contents

Chapter 1

Introduction

1.1 Introduction to Bioinformatics

Bioinformatics is an interdisciplinary field that develops methods and software tools for understanding biological data. As an interdisciplinary field of science, bioinformatics combines computer science, statistics, mathematics, and engineering to analyze and interpret biological data.

Bioinformatics has become a part of many areas of biology which of great importance. In experimental molecular biology, bioinformatics techniques such as image and signal processing allow extraction of useful results from large amounts of raw data. In the field of genetics and genomics, it aids in sequencing and annotating genomes and their observed mutations. It plays a role in the text mining of biological literature and the development of biological and gene ontologies to organize and query biological data. It also plays a role in the analysis of gene and protein expression and regulation. Bioinformatics tools aid in the comparison of genetic and genomic data and more generally in the understanding of evolutionary aspects of molecular biology.

Bioinformatics mainly involves inception, management and examination of biological data mainly obtained from a substantive number of experimental test runs that may at times provide large data set. To this effect, there is need to establish comprehensive mechanisms to aid in the interpretation and processing this data and consequently producing accurate information that is

needed for research and study purposes [1]. This led to the inception of bioinformatics, a discipline that integrates both biology and computer science.

The advances in biotechnology such as the next generation sequencing technologies are occurring at breathtaking speed. Advances and breakthroughs give competitive advantages to those who are prepared. However, the driving force behind the positive competition is not only limited to the technological advancement, but also to the companion data analytical skills and computational methods which are collectively called computational biology and bioinformatics. Without them, the biotechnology-output data by itself is raw and perhaps meaningless.[2]

1.2 Gene Expressions and Microarrays

"Gene expression is the term utilized for portraying the interpretation of data contained inside the DNA, the storehouse of hereditary data, into messenger RNA (mRNA) atoms that are then interpreted into the proteins that perform a large portion of the discriminating capacity of cells" [3]. Gene expression is a complex process that permits cells to respond to the changing inward prerequisites and furthermore to outside ecological conditions. This system controls which genes to express in a cell and furthermore to build or reduce the level of expression of genes.

A microarray is a device used to study and record the gene expressions of a large number of genes at the same time. A microarray comprises of distinctive nucleic acid probes that are artificially appended to a substrate, which can be a microchip, a glass slide or a microsphere-sized globule.

Microarrays can be divided into two types [4]:Dual Channel Microarrays and Single Channel Microarrays. The one we use in our experiment is the Single Channel Microarrays. In these microarrays, individual samples are subjected through hybridization after it is named with a

fluorescent color. These microarrays measure irrefutably the power of declaration. The last procedure is completed utilizing a methodology called photolithography.

In Dual Channel Microarrays, example successions and ordinary arrangements are marked with two distinctive fluorescent colors. Both these DNA arrangements are hybridized together on the DNA Microarrays and a degree of fluorescence intensities emitted by the two colors is considered so as to assess differential representation level.

There are distinctive sorts of microarrays, for example, DNA microarrays, protein microarrays, tissue microarrays and carb microarrays[5]. The microarray innovation was advanced out of the need to focus measures of specific substances inside a mixture of different substances.

It is possible to get gene expression data relatively inexpensively from micro-arrays. So, this leads to hope that the genes can tell us who will get or has a disease. Perhaps one can find the stage of the disease to enable effective treatments. However, we are currently at the stage where there are many challenges to evaluating the possibilities for genes to be used in diagnosis and treatment. There are typically many more genes that might be involved than samples for any given disease. Which genes are important and how stable are the choices an algorithm provides? We do not know the time of the true onset of a disease, just sometimes when symptoms started and sometimes when diagnosis was done. Some of the promising work on diagnosis or prognosis has suffered from data or scientific errors. [6] had discussed the problems and pitfalls of using genes to predict disease presence or prognosis. It had also discussed some promising ways to choose the genes that may be predictive for a particular disease, with a focus on cancer, and point out some open questions.

1.3 Classification Techniques

In this study, we use the classification technique to measure and compare the difference between different feature extraction/ selection methods. Any classification method uses a set of parameters to characterize each object. These features are relevant to the data being studied. Here we are talking about methods of supervised learning where we know the classes into which the items are to be ordered. We likewise have a set of items with known classes. A training set is utilized by the order projects to figure out how to arrange the items into wanted classes. This training set is utilized to choose how the parameters ought to be weighted or consolidated with one another so we can separate different classes of articles. In the application stage, the trained classifiers can be utilized to focus the classifications of articles utilizing new examples called the testing set.

Four classification methods viz Support Vector Machine, Random Forest, k-Nearest-neighbor, Neural Network have been selected in our study. The details about these methods are given in Chapter 3. We record the accuracy and AUC (Area under ROC) value for analysis.

1.4 Feature Extraction Methods and Feature Selection Methods

One big problems when applying large microarray dataset onto the different classifier is Redundancy. Some studies shown that the use of a feature extraction methods or feature selection method can somehow improve the performance and also reduce the redundancy. In our study, we will compare different performance between some well-known feature extraction methods (Principle Component Analysis, Partial Least Square) as well as some famous feature selection methods (minimum Redundancy-Maximum Relevance, Maximum Relevance,

Quadratic Programming Feature Selection). The detailed introduction of each method will be given in Chapter 4.

1.5 Objectives of the study and outline of the thesis

In this study, we proposed a measurement of vary feature preprocessing methods. This included the feature extraction method, feature selection method and discretized method. We are trying to find on what specific condition that can get the best performance among four different classifiers.

The thesis is organized as below:

We will present the literature review and some of the previous works done on the microarray classification. Plus, some results and works about feature extraction as well as feature selection methods will also be mentioned in Chapter 2.

The datasets and detailed classification methods will be describing in Chapter 3. Following the description and introduction of different feature extraction and selection method in Chapter 4.

Chapter 5 will give the expansion of the published paper about principle component analysis.

Another ongoing publish paper about the comparison of feature selection and extraction methods will be detailed in Chapter 6.

Conclusion, future work and other overall contents will list in and after Chapter 7.

Chapter 2

Literature Review

One big task in bioinformatics is to classify and predict the biological inferences of complex, high-dimensional and voluminous data. With the rapid increase in size and numbers of features in bioinformatics, it is critical to use some tools and good algorithms to automate the classification process. Several reports evidence that Classification methods viz SVM, NN, kNN and RF, have been applied in solving the bioinformatics problems with good accuracy.

Byvatov E , Schneider G noted that the Support Vector Machine(SVM) approach produced lower prediction error compared to classifiers based on other methods like artificial neural network, especially when large numbers of features are considered for sample description[7].[8] used a combination of SVM and Recursive feature elimination method in an experiment which showed that the genes selected by this method yield better classification performance and are biologically relevant to cancer. In [9], Yang discussed how the SVM can cooperate with biological data specifically in protein and DNA sequences. [10] showed that a neural network grows adopting the topology of a binary tree called Self-Organizing Tree Algorithm is a hierarchical cluster obtained with the accuracy and robustness of a neural network. They mentioned that this method is especially suitable for dealing with huge amounts of data. [11]presented a neural-network-based method to predict a given mutation increases or decreases the protein thermodynamic stability with respect to the native structure. They mentioned that one the same task and using the same data, their predictor performs better than other methods on the

web. [12]stated that random forest is a classification algorithm well suited for microarray data. They investigated the use of random forest for classification of microarray data and proposed a gene selection method in classification problems based on random forest. Also, in [13], Boulesteix, A , Janitza, S , synthesized the random forest development of 10 years with emphasis on applications to bioinformatics and computational biology. They showed that the RF algorithm has become a standard data analysis tool in bioinformatics. Besides the above, some researchers also using k-nearest neighbor for the similar tasks. [14] used the k-nearest neighbor based method for protein subcellular location prediction.[15] presented a learning approach which is derived from the traditional k-nearest-neighbor algorithm and gave a comparison on a multi-label bioinformatics data which showed that the new method is highly comparable to existing multi label learning algorithms. In [16], they compared another knn based method with Neural networks and SVM, and they concluded that the method is very promising and might become a useful vehicle in protein science, bioinformatics. In [17], Leping, L, tested an approach combines Genetic Algorithm and the k-nearest-neighbors to identify genes that can jointly discriminate between different classes of samples. They also chose the leukemia data set and the colon data set. The result showed that this method is capable of selecting a subset of predictive genes from a large noisy data set for sample classification.

Most of the works mentioned above focus on a specific classifier and made some improvement for the classification and even proposed some new methods which are driven by the classical methods. In our study, we want to know if the feature selection or feature extraction method can somehow improve the performance of gene selection problems. We will use the classical classification methods for the measurement. For the cross validation, [18] recommended using 10-fold cross validation rather than leave-one-out cross validation.

In classification tasks, the performance of the classifier improved when the dimensionality increases until an optimal number of feature is selected. After that, further increases of the dimensionality without increases of the samples will cause the performance degrade[19]. This is what we called the “curse of dimensionality”. To solve this problem, one way is to find out which subset of the origin data set is of vital importance for the classification tasks. In [20], a comparative study demonstrated that it is possible to use feature selection method to solve this problem. Authors have used SVM with different kernel and KNN method for a classification task and give a ranking method to two gene data set. The results showed the importance of informative gene ranking to classify test samples accurately. [21] present a hybrid method for feature selection. Their results showed that increased number of irrelevant and redundant features has decrease the accuracy of classifier as well as increase the computational cost and reinforced the curse of dimensionality. Hence feature extraction and feature selection method become a possible solution.

There are many different feature selection method, in [22], they presented a novel approach based on Binary Black Hole Algorithm and Random Forest Ranking. They use this method for gene selection and classification of microarray data. The results showed that selecting the least number of informative genes can increase prediction accuracy of Bagging and outperforms the other classification method.

When we move our attention to feature preprocessing methods, Peng H’s paper [23] gets our most attention, they proposed two methods namely mRMR and MaxRel for the gene selection and their experiments have showed that the methods are robust and stable for the improvement. In [24], they gave more detail and experiment to further investigate the two methods, the results

confirm that mRMR leads to promising improvement on feature selection and classification accuracy.

[25] have used PCA method for analysis of gene expression data. Their results showed that clustering with the PCA does not necessarily improve and often degrades the cluster quality. In [26], Antai W, proposed a gene selection method based on PCA and use this method to demonstrate that the method selects the best gene subset for preserving the original data structure. In our study, we will cover and give the results on classification tasks using PCA.

In [27], Jalali-Heravi M proposed a method which combined the genetic algorithm- kernel partial least square for feature selection. Their results indicated that this approach is a powerful method for the feature selection in nonlinear systems.

In [28], another feature selection method called Quadratic Programming Feature Selection is proposed. This method reduces the task to a quadratic optimization problem and is capable for dealing with very large data set. Plus, [29] proposed a sequential Minimal Optimization based framework for QPFS, and they successfully reduced the computation time of data dimension(cubic computation time) of standard QPFS.

In our study, we look forward to see how can the combination between the traditional classifiers and the well-performance feature selection method can somehow improve the performance. We covered PCA, MaxRel, PLS, QPFS, mRMR methods by using them for the preprocess process. After that, different validation methods were applied to the classification tasks. We give some suggestion of vary combination for new data sets.

Chapter 3

Datasets and Classification Methods

3.1 Dataset Selection

3.1.1 Colon Cancer Dataset

The Colon Cancer Dataset was first in use in [30, 31] in 1999. 62 samples, including 20 normal samples and 40 tumor samples, collected from patients of colon-cancer.

The matrix I2000 (See Appendix A: Matrix I2000 and Names of each Gene in Colon Dataset) contains the expression of the 2000 genes with highest minimal intensity across the 62 tissues. The genes are placed in order of descending minimal intensity. Each entry in I2000 is a gene intensity derived from the ~20 feature pairs that correspond to the gene on the chip, derived using the filtering process described in the 'materials and methods' section. The data is otherwise unprocessed (for example it has not been normalized by the mean intensity of each experiment).

The file 'names' contains the EST number and description of each of the 2000 genes, in an order that corresponds to the order in I2000. Note that some ESTs are repeated which means that they are tiled a number of times on the chip, with different choices of feature sequences.

The identity of the 62 tissues is given in file tissues. The numbers correspond to patients, a positive sign to a normal tissue, and a negative sign to a tumor tissue.

Before our experiments, we have normalized the dataset file to mean 0.

3.1.2 Leukemia Dataset

The Leukemia Dataset was also mentioned in [31]. Leukemias are primary disorders of bone marrow. They are malignant neoplasms of hematopoietic stem cells. The total number of genes to be tested is 7129, and number of samples to be tested is 72, which are all acute leukemia patients, either acute lymphoblastic leukemia (ALL) or acute myelogenous leukemia (AML).

More precisely, the number of ALL is 47 and the number of AML is 25. The dataset has been normalized before our experiments.

3.2 Classification Methods

In the current study, we deal with a classification problem which focuses on dividing the samples of two microarray datasets into two categories. Any classification method uses a set of parameters to characterize each object. These features are relevant to the data being studied. Here we are talking about methods of supervised learning where we know the classes into which the items are to be ordered. We likewise have a set of items with known classes. A training set is utilized by the order projects to figure out how to arrange the items into wanted classes. This training set is utilized to choose how the parameters ought to be weighted or consolidated with one another so we can separate different classes of articles. In the application stage, the trained

classifiers can be utilized to focus the classifications of articles utilizing new examples called the testing set. The different well-known to order techniques are discussed as follows [32].

3.2.1 Support Vector Machine

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. In 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes[33].The current standard incarnation (soft margin) was proposed by Corinna Cortes and Vapnik in 1993 and published in 1995.[34]

Let us look at the example for a clear understand.

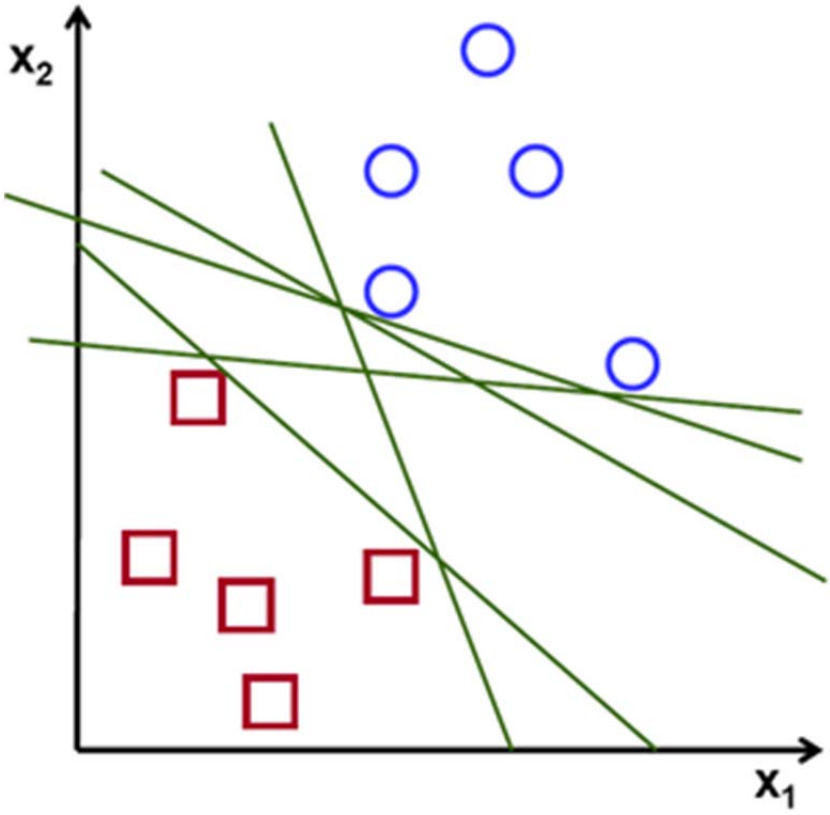

Figure 0-1: SVM Classification

Suppose we need to find a separating straight line for a linearly separable set of 2D-points which belong to one of two classes. In Figure 0-1, one can see that there exists multiple lines that offer a solution to the problem. Is any of them better than the others? The SVM define a criterion to estimate the worth of the lines:

A line is bad if it passes too close to the points because it will be noise sensitive and it will not generalize correctly. Therefore, our goal should be to find the line passing as far as possible from all points.

Then, the operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important

name of margin within SVM's theory. Therefore, the optimal separating hyperplane maximizes the margin of the training data. See Figure 0-2.

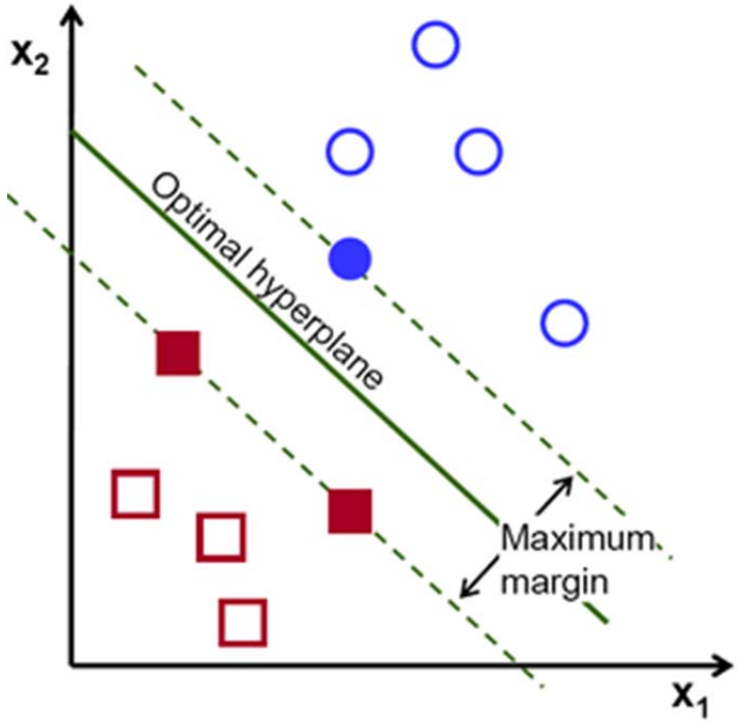

Figure 0-2: Optimal hyperplane

This is a simple example for SVM. For higher dimension situation, the same concepts apply to the task.

Formally a hyperplane can be defined as:

$$f(x) = \beta_0 + \beta^T x,$$

Where β is known as the weight vector and β_0 as the bias.

The optimal hyperplane can be represented in an infinite number of different ways by scaling of β and β_0. SVM would choose the one that meets:

$$|\beta_0 + \beta^T x| = 1,$$

where x symbolizes the training examples closest to the hyperplane. In general, the training examples that are closest to the hyperplane are called support vector. This representation is know as the canonical hyperplane.[35]

Now we use the result of geometry that gives the distance D between a point x and a hyperplane(β, β_0):

$$D = \frac{|\beta_0 + \beta^T x|}{\|\beta\|}.$$

Particularly, for canonical hyperplane, the numerator is equal to one and the distance to the support vector:

$$D_{SupportVector} = D = \frac{|\beta_0 + \beta_T x|}{\|\beta\|} = \frac{1}{\|\beta\|}$$

Recall that the margin introduced above, we denoted the margin as M, which is twice the distance to the closest examples:

$$M = \frac{2}{\|\beta\|}$$

Last but not least, the problem of maximizing M is equivalent to the problem of minimizing a function $L(\beta)$ subject to some constraints. The constraints model the requirement for the hyperplane to classify correctly all the training examples x_i. Formally:

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}\|\beta\|^2$$

subject to:

$$y_i(\beta^T x_i + \beta_0) \geq 1 \; \forall i,$$

where $y_i$represents each of the labels of the training examples.[36]

In [37], Jin, Chi; Wang, Liwei mentioned that for a higher-dimensional feature space, the generalization error of SVM still performs well even if we give enough samples.

The SVM function we use in the experiment is available as a plugin in Weka. The original wrapper, named WLSVM, was developed by Yasser EL-Manzalawy. The current version we use is complete rewrite of the wrapper in order to avoid compilation errors. Package name and all the parameters will be given in 3.2.5.

3.2.2 Random Forest

The first algorithm for random decision forests was created by Tin Kam Ho [38] using the random subspace method,[39] which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.[40, 41]

The function we use in the experiment is available in Weka. The function fully implemented the method in [42]. The development, verification and the significance of variable importance of RF were focused in [43].

The RF is an ensemble approach that can also be thought of as a form of nearest neighbor predictor. This will be mentioned in 3.2.4.

Ensembles are a divide-and-conquer approach used to improve performance. The main target of the ensemble methods is that a group of weak learners can come together to form a strong learner.

Here is an example from [44], in Figure 0-3, the data to be modeled are the blue circles. We assume that they represent some underlying function plus noise. Each individual learner is shown as a gray curve. Each gray curve (a weak learner) is a fair approximation to the underlying data. The red curve (the ensemble "strong learner") can be seen to be a much better approximation to the underlying data.

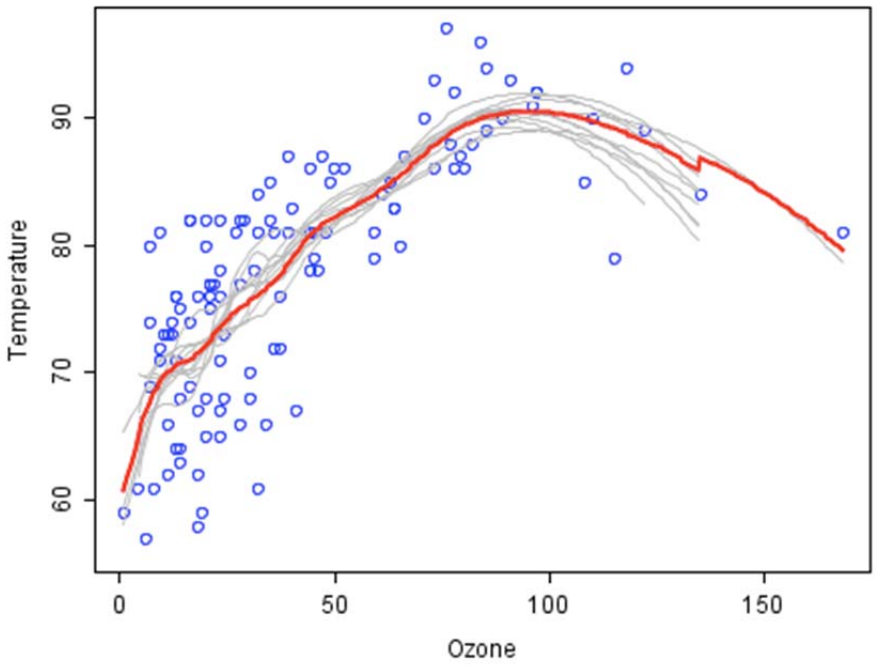

Figure 0-3: Strong learner and Weak learner

The random forest starts with a standard machine learning technique called a "decision tree" which, in ensemble terms, corresponds to our weak learner. In a decision tree, an input is entered at the top and as it traverses down the tree the data gets bucketed into smaller and smaller sets.

In this example, see Figure 0-4 ,the tree advises us, based upon weather conditions, whether to play ball. For example, if the outlook is sunny and the humidity is less than or equal to 70, then it's probably OK to play.

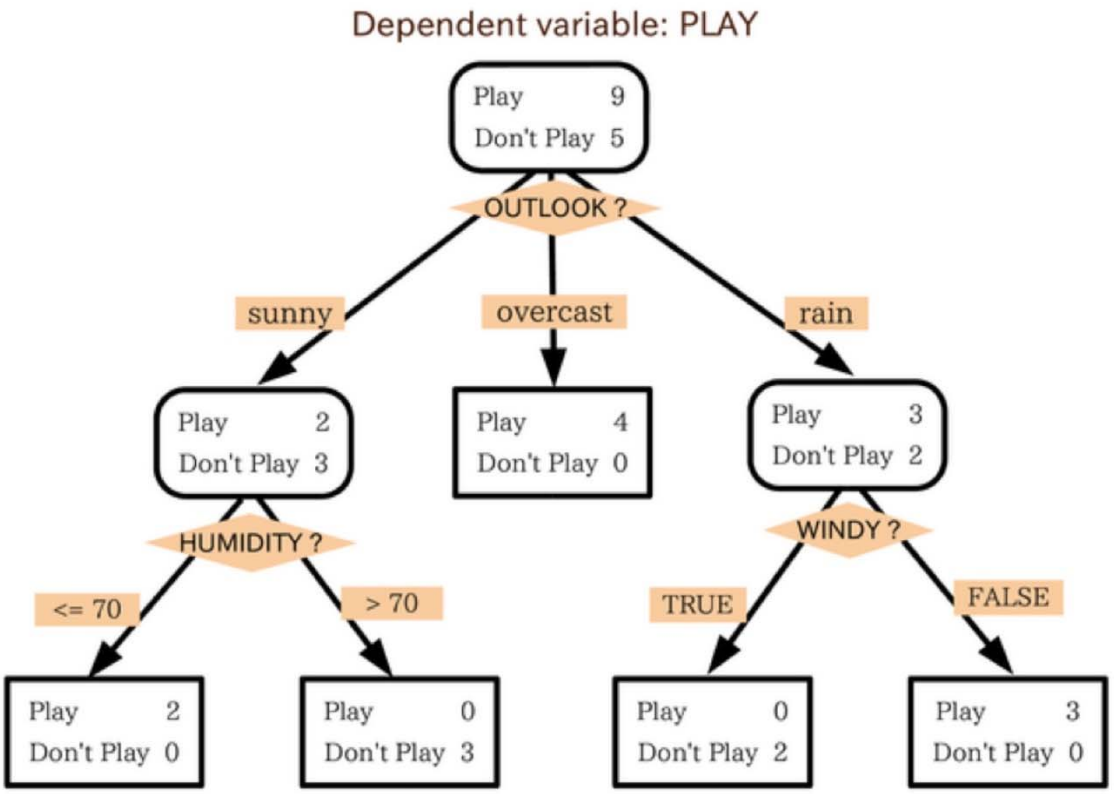

Figure 0-4: Example of a Decision Tree

The random forest takes this notion to the next level by combining trees with the notion of an ensemble (See Figure 0-5). Thus, in ensemble terms, the trees are weak learners and the random forest is a strong learner.

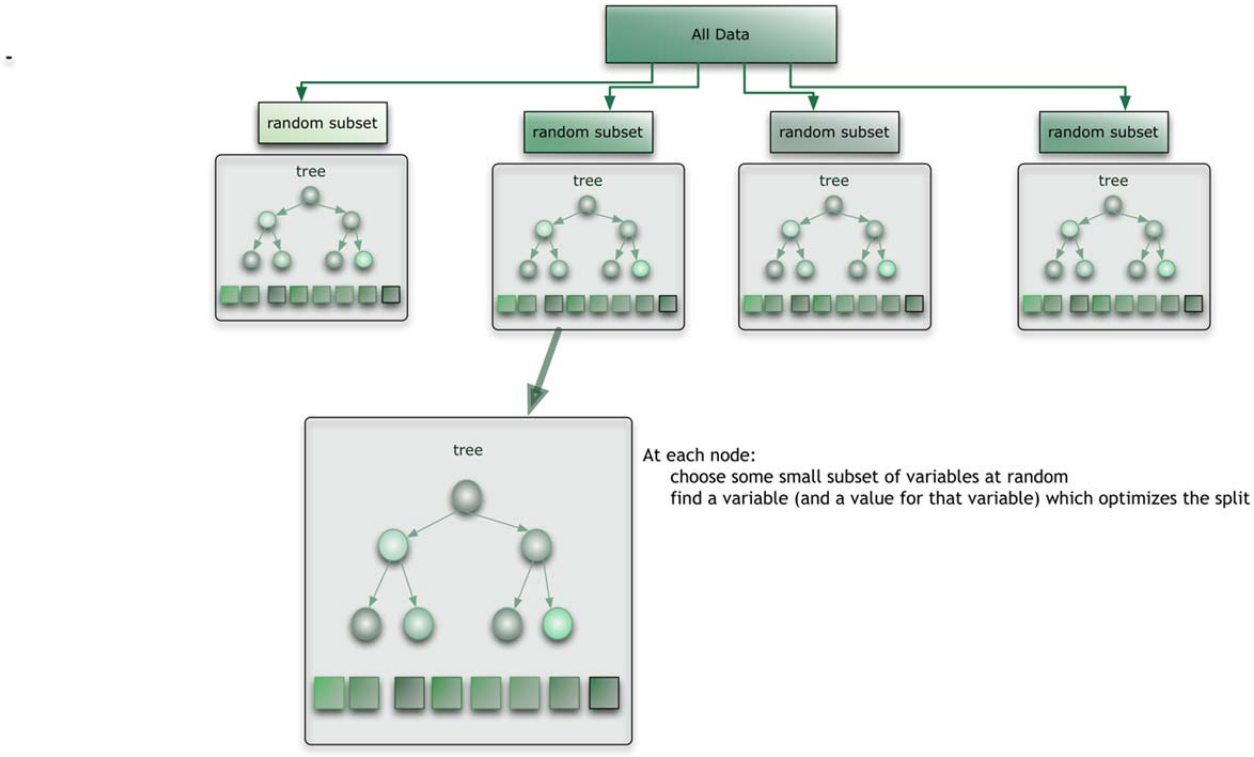

Figure 0-5: Relationship between Random Forest and Random Tree

Here is how such a system is trained; for some number of trees T:

Sample N cases at random with replacement to create a subset of the data (see Figure 0-5). The subset should be about 66% of the total set.

At each node:

For some number m (see below), m predictor variables are selected at random from all the predictor variables.

The predictor variable that provides the best split, according to some objective function, is used to do a binary split on that node.

At the next node, choose another m variables at random from all predictor variables and do the same.

Depending upon the value of m, there are three slightly different systems:

Random splitter selection: m =1

Breiman's bagger: m = total number of predictor variables

Random forest: m << number of predictor variables. Brieman suggests three possible values for m: $\frac{1}{2}\sqrt{m}, \sqrt{m}, 2\sqrt{m}$.

When a new input is entered into the system, it is run down all of the trees. The result may either be an average or weighted average of all of the terminal nodes that are reached, or, in the case of categorical variables, a voting majority.

Random forest runtimes are quite fast, and they are able to deal with unbalanced and missing data. Random Forest weaknesses are that when used for regression they cannot predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy.

The parameters that we use in the experiment are given in 3.2.5.

3.2.3 Neural network

Inspired by the sophisticated functionality of human brains where hundreds of billions of interconnected neurons process information in parallel, researchers have successfully tried demonstrating certain levels of intelligence on silicon. Examples include language translation and pattern recognition software. While simulation of human consciousness and emotion is still in the realm of science fiction.

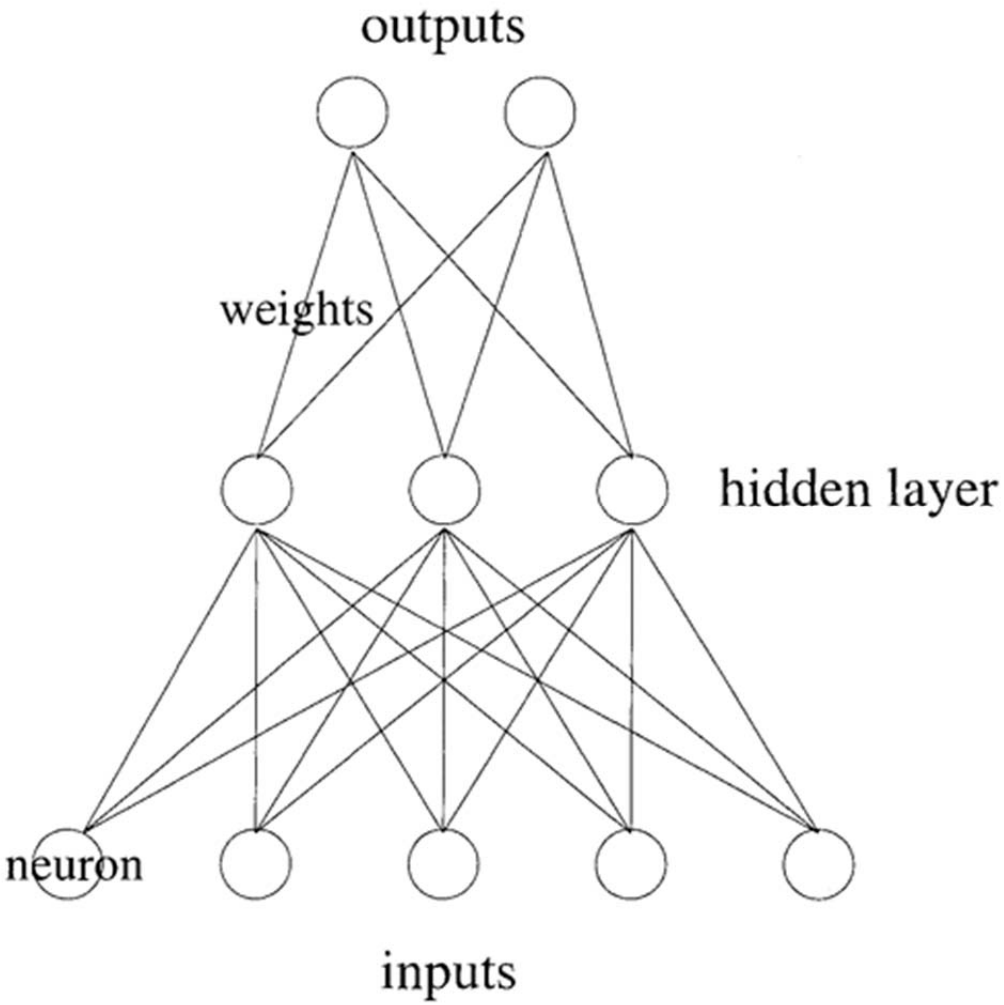

Figure 0-6: Architecture of Neural Network

An artificial neural network(NN) consists of an input layer of neurons (nodes, units), one or two (or even three) hidden layers of neurons, and a final layer of output neurons. See Figure 0-6 for a typical architecture, where lines connecting neurons are also shown. Each connection is associated with a numeric number called weight. The output, h_i, of neuron i in the hidden layer is:

$$h_i = \sigma(\sum_{j=1}^{N} V_{ij} x_j + T_i^{hid}),$$

where $\sigma()$ is called activation function, N the number of input neurons, V_{ij} the weights, x_j inputs to the input neurons, and T_i^{hid} the threshold terms of the hidden neurons.[45]

The most straightforward structural planning of fake neural systems is single-layered system, likewise called Perceptron, where inputs connect directly to the outputs through a single layer of weights. The most usually utilized type of NN is the Multilayer Perceptron. NN offers a compelling and exceptionally general structure for speaking to non-linear mapping from a few information variables to a few yield variables.[46]

We will use the Weka in-built function 'MultilayerPerceptron' for the experiment. Detailed parameter will be given in 3.2.5.

3.2.4 k-Nearest-Neighbor

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.[47, 48]

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If $K = 1$, then the case is simply assigned to the class of its nearest neighbor.

The well-known distance functions are Euclidean Distance, Manhattan Distance, Minkowski Distance and Hamming Distance.

$$\text{Euclidean: } \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan: $\sum_{i=1}^{k} |x_i - y_i|$

Minkowski: $\left(\sum_{i=1}^{k} (|x_i - y_i|)^q\right)^{\frac{1}{q}}$

Hamming: $D = \sum_{i=1}^{k} |x_i - y_i|$, where $x = y \Rightarrow D = 0; x \neq y \Rightarrow D = 1$

It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables, the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

An example in Figure 0-7, The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If $K = 3$ (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If $K = 5$ (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).

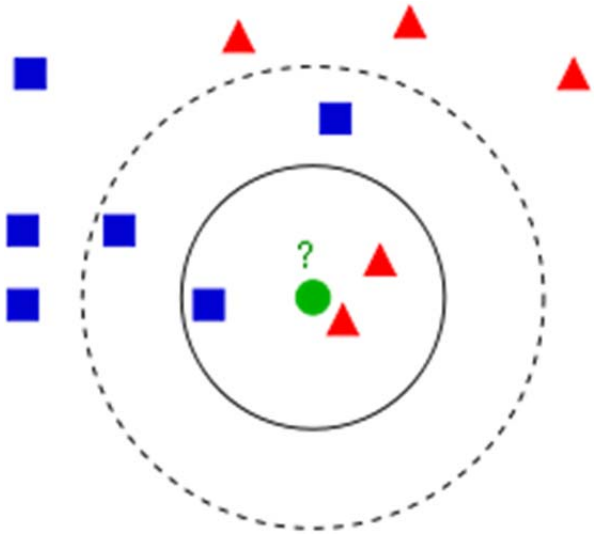

Figure 0-7: k-NN circle with k=3 and k=5

In the classification phase, K is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the K training samples nearest to that query point.

Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.[49]

In our experiment, we will use $K = 1$ and the Euclidean Distance.

3.2.5 List of Classification Parameters

Classifier	Weka Function	Parameters
SVM	LibSVM	-S 0 -K 0 -D 3 -G 0.0 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.001 -P 0.1 -B -model /Users/Stanley -seed 1
RF	RandomForest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
NN	MultilayerPerceptron	-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
KNN	IBK	-K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Chapter 4

Feature Extraction Method and Feature Selection Methods

The task of the feature extraction and selection methods is to get the most relevant information from the original data and represent that information in a lower dimensionality space.[50]

For the different between feature extraction and feature selection, feature selection is to select the relevant information from the original data without modify the original attributes. Feature selection gets the new data set by use the subset of original data set. Feature extraction is to select the information and use the information from original attributes to combine for the new data set.

Dimensionality reduction is typically choosing a basis or mathematical representation within which you can describe most but not all of the variance within your data, thereby retaining the relevant information, while reducing the amount of information necessary to represent it. There are a variety of techniques for doing this including but not limited to PCA, ICA, and Matrix Feature Factorization. These will take existing data and reduce it to the most discriminative components. These all allow you to represent most of the information in your dataset with fewer, more discriminative features.

Feature Selection is hand selecting features which are highly discriminative. This has a lot more to do with feature engineering than analysis, and requires significantly more work on the part of the data scientist. It requires an understanding of what aspects of your dataset are important in whatever predictions you're making, and which aren't. Feature extraction usually involves

generating new features which are composites of existing features. Both of these techniques fall into the category of feature engineering. Generally, feature engineering is important if you want to obtain the best results, as it involves creating information that may not exist in your dataset, and increasing your signal to noise ratio.

4.1 Feature Extraction Method

The goal is to build, using the available features, those that will perform better. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy.

In our experiment, we choose Principle Component Analysis and Partial Least Square as the feature extraction methods.

Detailed introduction will be given in 4.3.1 and 4.3.2.

4.2 Feature Selection Methods

Feature selection is also called variable selection or attribute selection.

It is the automatic selection of attributes in your data (such as columns in tabular data) that are most relevant to the predictive modeling problem you are working on.

Feature selection is different from dimensionality reduction. Both methods seek to reduce the number of attributes in the dataset, but a dimensionality reduction method do so by creating new combinations of attributes, whereas feature selection methods include and exclude attributes present in the data without changing them.[51]

Feature selection methods can be used to identify and remove unneeded, irrelevant and redundant attributes from data that do not contribute to the accuracy of a predictive model or may in fact decrease the accuracy of the model.

There are three general classes of feature selection algorithms: filter methods, wrapper methods and embedded methods.

Filter feature selection methods apply a statistical measure to assign a scoring to each feature. The features are ranked by the score and either selected to be kept or removed from the dataset. The methods are often univariate and consider the feature independently, or with regard to the dependent variable.

Example of some filter methods include the Chi squared test, information gain and correlation coefficient scores.

Wrapper methods consider the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. A predictive model us used to evaluate a combination of features and assign a score based on model accuracy.

The search process may be methodical such as a best-first search, it may stochastic such as a random hill-climbing algorithm, or it may use heuristics, like forward and backward passes to add and remove features.

An example if a wrapper method is the recursive feature elimination algorithm.

Embedded methods learn which features best contribute to the accuracy of the model while the model is being created. The most common type of embedded feature selection methods are regularization methods.

Regularization methods are also called penalization methods that introduce additional constraints into the optimization of a predictive algorithm (such as a regression algorithm) that bias the model toward lower complexity (less coefficients).

Examples of regularization algorithms are the LASSO, Elastic Net and Ridge Regression.

4.3 Methods in Our Experiment

4.3.1 Principal Component Analysis(PCA)

Principal Component Analysis was invented in 1901.[52] as an analogue of the principal axis theorem in mechanics; it was later independently developed (and named) by Harold Hotelling in the 1930s.[53] Depending on the field of application, it is also named the discrete Kosambi-Karhunen–Loève transform (KLT) in signal processing, the Hotelling transform in multivariate quality control, proper orthogonal decomposition (POD) in mechanical engineering, singular value decomposition (SVD) of X [54], eigenvalue decomposition (EVD) of XTX in linear algebra, factor analysis (for a discussion of the differences between PCA and factor analysis see Ch. 7 of [55]).

PCA is mostly used as a tool in exploratory data analysis and for making predictive models. PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of a data matrix, usually after mean centering (and normalizing or using Z-scores) the data matrix for each attribute.[56] The results of a PCA are usually discussed in terms of component scores, sometimes called factor scores (the transformed variable values

corresponding to a particular data point), and loadings (the weight by which each standardized original variable should be multiplied to get the component score).[57]

The most common definition of PCA, in [53], is that, for a given set of data vectors x_i , $i \in 1 \ldots t$, the d principal axes are those orthonormal axes onto which the variance retained under projection is maximal.

In order to capture as much of the variability as possible, let us choose the first principal component, denoted by U_1, to have maximum variance. Suppose that all centered observations are stacked into the columns of an $n \times t$ matrix X, where each column corresponds to an n-dimensional observation and there are t observations. Let the first principal component be a linear combination of X defined by coefficients (or weights) $w = [w_1 \ldots w_n]$.

In matrix form:

$$U_1 = w^T X$$

$$var(U_1) = var(w^T X) = w^T S w$$

where S is the $n \times n$ sample covariance matrix of X. Clearly $var(U_1)$ can be made arbitrarily large by increasing the magnitude of w. Therefore, we choose w to maximize $w^T S w$ while constraining w to have unit length.

$$max\ w^T S w$$

subject to

$$w^T w = 1$$

To solve this optimization problem, a Lagrange multiplier α_1 is introduced:

$$L(w, \alpha) = w^T S w - \alpha_1 (w^T w - 1)$$

Differentiating with respect to w gives n equations,

$$Sw = \alpha_1 w$$

Premultiplying both sides by w^T we have:

$$w^T S w = \alpha_1 w^T w = \alpha 1$$

$var(U_1)$ is maximized if α_1 is the largest eigenvalue of S.

Clearly α_1 and w are an eigenvalue and an eigenvector of S. Differentiating $L(w, \alpha)$ with

respect to the Lagrange multiplier α_1 gives us back the constraint:

$$w^T w = 1$$

This shows that the first principal component is given by the normalized eigenvector with the largest associated eigenvalue of the sample covariance matrix S. A similar argument can show that the d dominant eigenvectors of covariance matrix S determine the first d principal components.

Another nice property of PCA, closely related to the original discussion by Pearson in[52] is that the projection onto the principal subspace minimizes the squared reconstruction error,

$$\Sigma_{i=1}^{t} \|x_i - \hat{x}_i\|^2$$

In other words, the principal components of a set of data in R^N provide a sequence of best linear approximations to that data, for all ranks $d \leq n$. Consider the rank-d linear approximation model as:

$$f(y) = \overline{x} + U_d y$$

This is the parametric representation of a hyperplane of rank d. For convenience, suppose $\overline{x} = 0$ (otherwise the observations can be simply replaced by their centered versions $\tilde{x} = x_i - \overline{x}$). Under this assumption the rank d linear model would be $f(y) = U_d y$, where U_d is a $n \times d$ matrix with d orthogonal unit vectors as columns and y is a vector of parameters. Fitting this model to the data by least squares leaves us to minimize the reconstruction error:

$$\min_{U_d, y_i} \Sigma_i^t \|x_i - U_d y_i\|^2,$$

By partial optimization for y_i we obtain:

$$\frac{d}{dy_i} = 0 \Rightarrow y_i = U_d^T x_i$$

Now we need to find the orthogonal matrix U_d:

$$\min_{U_d} \Sigma_i^t \|x_i - U_d U_d^T x_i\|^2$$

Define $H_d = U_d U_d^T$. H_d is a $n \times n$ matrix which acts as a projection matrix and projects each data point x_i onto its rank d reconstruction. In other words, $H_d x_i$ is the orthogonal projection of x_i onto the subspace spanned by the columns of U_d. A unique solution U can be obtained by finding the singular value decomposition of X [35]. For each rank d, U_d consists of the first d

columns of U. Clearly the solution for U can be expressed as singular value decomposition (SVD) of X.

$$X = U\Sigma V^T$$

since the columns of U in the SVD contain the eigenvectors of XX^T. So the PCA procedure is summarized in Table 0-1.

Table 0-1: PCA Algorithm

Algorithm of PCA

1. Recover basis:

   Calculate $XX^T = \Sigma_{i=1}^{t} x_i x_i^T$ and let $U = eigenvectors\ of\ XX^T$

   corresponding to the top d eigenvalues.
2. Encode training data: $Y = U^T X$ where Y is a $d \times t$ matrix of encodings of the original data.
3. Reconstruct training data: $\hat{X} = UY = UU^T X$.
4. Encode test example: $y = U^T x$ where y is a d-dimensional encoding of x.
5. Reconstruct test example: $\hat{x} = U_y = UU^T x$

We will use the Weka Principle Component function in our experiment. The function is to performs a principal components analysis and transformation of the data. Use in conjunction with a Ranker search. Dimensionality reduction is accomplished by choosing enough eigenvectors to account for some percentage of the variance in the original data---default 0.95 (95%). Attribute noise can be filtered by transforming to the PC space, eliminating some of the worst eigenvectors, and then transforming back to the original space.

4.3.2 Partial Least Square (PLS)

Partial least squares regression (PLS regression) is a statistical method that bears some relation to principal components regression; instead of finding hyperplanes of maximum variance between the response and independent variables, it finds a linear regression model by projecting the predicted variables and the observable variables to a new space.

Partial Least Squares is a simultaneous feature extraction and regression technique, well suited for high dimensional problems where the number of samples is much lesser than the number of features ($n \ll p$). The linear PLS model can be expressed as:

$$X = TP^{\top} + X_{res}$$

$$Y = UQ^{\top} + Y_{res}$$

where $X_{n\times p}$ is the feature matrix, $Y_{n\times q}$ is the matrix of response variables or class labels, $T_{n\times d}$ is called the X-scores, $P_{p\times d}$ is X-loading, $U_{n\times d}$ is Y-scores, $Q_{q\times d}$ is Y-loadings, X_{res} and Y_{res} are the residuals. The data in X and Y are assumed to be mean-centered. X-scores and Y-scores are the projections of n samples onto a d-dimensional orthogonal subspace. The X-scores are obtained by a linear combination of the variables in X with the weights W as:

$$T = XW^{*}$$

The inner relation between X-scores and Y-scores is a linear regression model and hence X-scores are called predictors of Y-scores. If we denote B as the regression coefficient for the inner relation between the scores, we have:

$$U = TB$$

So that we get:

$$Y = UQ^{\top} + Y_{res}$$
$$= TBQ^{\top} + Y_{res}$$
$$= T\overline{B} + Y_{res}$$

where $\overline{B} = BQ^{\top}$

The least squares estimate of $\overline{B}$ is then given by:

$$\hat{B} = (T^{\top}T)^{-1}T^{\top}Y$$

Hence PLS can be expressed in a linear regression from:

$$\hat{Y} = T\hat{B} = T(T^{\top}T)^{-1}T^{\top}Y$$

For more detail about the explanation of the PLS, see [58-60].

The two most popular algorithms to obtain the PLS model are NIPALS [80] and SIMPLS [81]. We have implemented SIMPLS available in Weka.

4.3.3 Peng's MaxRel Method and mRMR Method

mRMR means minimum-Redundancy-Maximum-Relevance feature/variable/attribute selection. The goal is to select a feature subset set that best characterizes the statistical property of a target classification variable, subject to the constraint that these features are mutually as dissimilar to each other as possible, but marginally as similar to the classification variable as possible. There

are several different forms of mRMR, where "relevance" and "redundancy" were defined using mutual information, correlation, t-test/F-test, distances, etc.

Importantly, for mutual information, they showed that the method to detect mRMR features also searches for a feature set of which features jointly have the maximal statistical "dependency" on the classification variable. This "dependency" term is defined using a new form of the high-dimensional mutual information.

The mRMR method was first developed as a fast and powerful feature "filter". Then they also showed a method to combine mRMR and "wrapper" selection methods. These methods have produced promising results on a range of datasets in many different areas.[24]

4.3.3.1 Discretization Preprocessing Method

Before we go through the MaxRel and mRMR methods, we will introduce the discretize method in our experiment.

Discretization is an essential pre-processing step for machine learning algorithms that can handle only discrete data. However, discretization can also be useful for machine learning algorithms that directly handle continuous variables. [61] indicated that the improvement in classification performance from discretization accrues to a large extent from variable selection and to a smaller extent from the transformation of the variable from continuous to discrete.

Many studies [62-66]have shown that induction tasks can benefit from discretization: rules with discrete values are normally shorter and hence easier to understandable and discretization can lead to improved predictive accuracy.

[23, 24] showed that discretization will often lead to a more robust classification. There are several ways for the discretization. Such as Binary discretization. Each feature variable was divided at the mean value, the value become 1 if it is large than the mean value and -1 otherwise. Another 3-states discretization method which we will use in our experiment is we use

$$\mu \pm \sigma$$

as the divided point, where μ is the mean value and σ is the standard deviation.

The value become -1 if it is less than $\mu - \sigma$,

1 if it is larger than $\mu + \sigma$, and 0 if otherwise.

We use python to convert this discretization. See Table 0-1 for codes.

4.3.3.2 Max Relevant Feature Selection (MaxRel)

One of the most popular approaches to realize Max-Dependency is Maximal Relevance (MaxRel) feature selection: selecting the features with the highest relevance to the target class c. Relevance is usually characterized in terms of correlation or mutual information, of which the latter is one of the widely used measures to define dependency of variables. Given two random variables x and y, their mutual information is defined in terms of their probabilistic density functions $p(x), p(y), and\ p(x, y)$:

$$I(x, y) = \iint p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dxdy$$

In MaxRel, the selected features x_i are required, individually, to have the largest mutual information $I(x_i, c)$ with the target class c, reflecting the largest dependency on the target class.

In terms of sequential search, the m best individual features, i.e., the top m features in the descent ordering of $I(x_i, c)$ are often selected as the m features.

To measure the level of discriminant powers of genes when they are differentially expressed for different target classes, we again use mutual information $I(h, g_i)$ between targeted classes $h = (h_1, h_2, \ldots, h_K)$ (we call h the classification variable) and the gene expression g_i. $I(h, g_i)$ quantifies the relevance of g_i for the classification task. Thus the maximum relevance condition is to maximize the total relevance of all genes in the subset we are seeking (Denoted as S):

$$maxV_I, V_I = \frac{1}{|S|} \Sigma_{i \in S} I(h, g_i),$$

With what we mentioned in 4.3.3.1, we denote the MaxRel feature selection method with continuous data as MRC method and MRD for discretized data.

4.3.3.3 Minimum Redundant – Maximum Relevant (mRMR)

It is likely that features selected according to Max-Relevance could have rich redundancy, i.e., the dependency among these features could be large. When two features highly depend on each other, the respective class-discriminative power would not change much if one of them were removed. Therefore, the following minimal Redundancy (mR) condition can be added to select mutually exclusive features:

$$minW_I, W_I = \frac{1}{|S|^2} \Sigma_{i,j \in S} I(g_i, g_j)$$

The mRMR feature selection method is obtained by balance the Maximum Relevant and Minimum Redundant conditions. Optimization of both conditions requires combining them into

a single criterion function. The method where the author of mRMR treat the two conditions equally important, and consider two simplest combination criteria:

$$\max(V_I - W_I)$$

$$\max\left(\frac{V_I}{W_I}\right)$$

These two ways to combine relevance and redundancy lead to the selection criteria of a new feature. We can choose either Mutual Information Difference criterion(MID) or Mutual Information Quotient criterion(MIQ).

In our experiment, we will cover both MID and MIQ method. We denote the 4 different methods as Table 0-2.

Table 0-2:Four methods from mRMR

Methods	Mutual Information Criterion	Data type
mRDD	MID	Discrete
mRDQ	MIQ	Discrete
mRCD	MID	Continuous
mRCQ	MIQ	Continuous

4.3.3.4 Mutual information estimation

We consider mutual-information-based feature selection for both discrete and continuous data. For discrete (categorical) feature variables, the integral operation in mutual information between two variables reduces to summation. In this case, computing mutual information is straightforward, because both joint and marginal probability tables can be estimated by tallying the samples of categorical variables in the data.

However, when at least one of variables x and y is continuous, their mutual information $I(x, y)$ is hard to compute, because it is often difficult to compute the integral in the continuous space based on a limited number of samples. One solution is to incorporate data discretization as a preprocessing step. For some applications where it is unclear how to properly discretize the continuous data, an alternative solution is to use density estimation method (e.g., Parzen windows) to approximate $I(x, y)$, as suggested by earlier work in medical image registration [67] and feature selection [68].

Given N samples of a variable x, the approximate density function $\hat{p}(x)$has the following form:

$$\hat{p}(x) = \frac{1}{N} \Sigma_{i=1}^{N} \delta\left(x - x^{(i)}, h\right),$$

where $\delta()$ is the Parzen window function as explained below, $x^{(i)}$ is the ith sample, and h is the window width. Parzen has proven that, with the properly chosen $\delta()$, and h, the estimation $\hat{p}(x)$ can converge to the true density $p(x)$ when N goes to infinity.[69]. Usually, $\delta()$,is chosen as the Gaussian window:

$$\delta(z,h) = \frac{\exp\left(-\frac{z^T \Sigma^{-1} z}{2h^2}\right)}{(2\pi)^{\frac{d}{2}} h^d |\Sigma|^{\frac{1}{2}}}$$

where $z = x - x^{(i)}$, d is the dimension of the sample x and Σ is the covariance of z. When $d = 1$, $\hat{p}(x)$ returns the estimated marginal density. When $d = 2$, $\hat{p}(x)$ can be use to estimate the density of bivariate variable $(x, y), p(x, y)$, which is actually the joint densty of x and y. For the sake of robust estimation, for $d \geq 2, \Sigma$ is often approximated by its diagonal components.

4.3.4 Quadratic Programming Feature Selection (QPFS)

Quadratic Programming Feature Selection was first introduced in [28], the target for this method is to reduce the feature selection task to a quadratic optimization problem. This method uses the Nystrom method[70] for approximate matrix diagonalization. This method is ideal for handle very large data sets for which the use of other methods is computationally expensive.

Assume the classifier learning problem involves N training samples and M variables (also called attributes or features). A quadratic programming problem is to minimize a multivariate quadratic function subject to linear constraints as follows:

$$\min_x (\frac{1}{2} x^T Q x - F^T x)$$

where x is an M-dimensional vector, $Q \in R^{M \times M}$is a symmetric positive semidefinite matrix, and F is a vector in $R^M$with non-negative entries. Applied to the feature selection task, Q represents the similarity among variables(Redundancy), and F measures how correlated each feature is with the target class(Relevance).

If the quadratic programming optimization problem has been solved, the components of x represent the weight of each feature. Features with higher weights are better variables to use for subsequent classifier training. Since x_i represents the weight of each variable, it is reasonable to enforce the following constraints:

$$x_i \geq 0 \; \forall i = 1, \ldots, M$$

$$\Sigma_{i=1}^{M} x_i = 1$$

Depending on the learning problem, the quadratic and linear terms can have different relative purposes in the objective function. Therefore, we introduce a scalar parameter α as follows:

$$\min_x (\frac{1}{2}(1-\alpha)x^T Q x - \alpha F^T x)$$

where $\alpha \in [0,1]$, if $\alpha = 0.5$, this problem is then equal to mRMR method.

QPFS using mutual information as its similarity measure resembles mRMR, but there is an important difference. The mRMR method selects features greedily, as a function of features chosen in previous steps. In contrast, QPFS is not greedy and provides a ranking of features that takes into account simultaneously the mutual information between all pairs of features and the relevance of each feature to the class label.

In our experiment, we use the program from the author’s code repository. Detailed parameter and environment will be given in 4.3.5.

4.3.5 List of Parameters of PCA, PLS and Feature Selection Methods

Different platforms are used to execute various programs; the details of platform and settings of parameters are listed in Table 0-3. PCA and PLS are in-built functions in WEKA. MaxRel and mRMR have been implemented using the source code from the author's website and was compiled on MAC OS 10.11.6. QPFS was implemented using the source code from a related google code repository and compiled on Ubuntu due to the convenience of installing reliable computational package.

Table 0-3:Parameters of Methods

Method	Platform/Software	Parameter	Comment
PCA	Weka 3-7-13-oracle-jvm MAC OS 10.11.1	weka.filters.unsupervised. attribute.PrincipalComponents -R [range] -A 5 -M -1	We choose the subset of PCA from range in [0.5,0.55,0.6,0.65,0.7,0.75,0.8, 0.85,0.9,0.95]
PLS	Weka 3-9-0-oracle-jvm MAC OS 10.11.6	weka.filters.supervised. attribute.PLSFilter -C [range] -M -A PLS1 -P center	We choose the subset of PLS from range in [5,10,20,30,50,100,200]
MaxRel mRMR	http://penglab.janelia.org /proj/mRMR/ MAC OS 10.11.6	./mrmr –i [Datasets] –n [range] –m [method]	range in [5,10,20,30,50,100,200] method in [MIQ,MID]
QPFS	https://sites.google.com/site/irenerodriguezlujan/documents/qpfs Ubuntu 14.04 LTS	./QPFS –F [Datasets] –O output.txt	The QPFS gave all the features a rank, we than select the top range features to build the subsets. range in [5,10,20,30,50,100,200]

Chapter 5

Increasing Efficiency of Microarray Analysis by PCA[1]

Principal Component Analysis (PCA) is widely used method for dimensionality reduction. However, it has not been studied much as a feature selection method to increase the efficiency of the classifiers on microarray data analysis. In this chapter, we assessed the performance of four classifiers on the microarray datasets of colon and leukemia cancer before and after applying PCA as a feature selection method. Different thresholds were used with 10-fold cross validation. Significant improvement was observed in the performance of the well-known machine learning classifiers on microarray datasets of colon and leukemia after applying PCA.

The gene expression profiling techniques by DNA microarrays provide the analysis of large amount of genes [71]. The number of gene expression data of microarray has grown exponentially. It is of great importance to find the key gene expression which can best describe the phenotypic trait [72]. The microarray dataset usually has a large number of genes in small number of experiments which collectively raise the issue of "curse of dimensionality"[73]. To find the key gene expression, one way is to use feature selection methods. In this chapter, we use

[1] J. Sun, K. Passi and C.K. Jain, Increasing Efficiency of Microarray Analysis by PCA and Machine Learning Methods, The 17th International Conference on Bioinformatics & Computational Biology (BIOCOMP'16), in The 2016 World Congress in Computer Science, Computer Engineering & Applied Computing, July 25 – 28, 2016, Las Vegas, Nevada, USA.

Principal Component Analysis (PCA) for feature selection and apply four well-known machine learning methods, Support Vector Machine (SVM), Neural Network (NN), K-Nearest-Neighbor (KNN) and Random Forest algorithms to validate and compare the performance of Principal Component Analysis. In the first set of experiments presented in this chapter, the performance of the four machine learning techniques (SVM, NN, KNN, Random Forest) is compared on the colon and leukemia microarray datasets. The second set of experiments compares the performance of these machine learning algorithms by applying PCA method on the same datasets.

5.1 Tools

In this experiment, Weka is the main testing tools for either 10-folds cross validation as well as the ratio comparison. The version we use in this experiment is 3-7-13-oracle-jvm and can be download on its official website. The platform we use in this experiment is Mac OS 10.11.1.

The Weka KnowledgeFlow Environment presents a "data-flow" inspired interface to Weka. The user can select Weka components from a tool bar, place them on a layout canvas and connect them together in order to form a "knowledge flow" for processing and analyzing data. At present, all of Weka's classifiers and filters are available in the KnowledgeFlow along with some extra tools.

We use Weka KnowledgeFlow for the 10-folds cross validation and Ratio Comparison. The flow chart will be given in 5.3 below and 5.4 below.

5.2 PCA

Principal Component Analysis (PCA) is a multivariate technique that analyzes a data table in which observations are described by several inter-correlated quantitative dependent variables. Its goal is to extract the important information from the table, to represent it as a set of new orthogonal variables called principal components, and to display the pattern of similarity of the observations and of the variables as points in maps. The quality of the PCA model can be evaluated using cross-validation techniques. Mathematically, PCA depends upon the eigen-decomposition of positive semi-definite matrices and upon the singular value decomposition (SVD) of rectangular matrices [56]. The PCA viewpoint requires that one compute the eigenvalues and eigenvectors of the covariance matrix, which is the product XX^T , where X is the data matrix. Since the covariance matrix is symmetric, the matrix is diagonalizable, and the eigenvectors can be normalized such that they are orthonormal:

$$XX^T = WDW^T$$

On the other hand, applying SVD to the data matrix X as follows:

$$X = U\Sigma V\text{^}T$$

and attempting to construct the covariance matrix from this decomposition gives

$$XX^T = (U\Sigma V^T)(U\Sigma V^T)^T$$

$$XX^T = (U\Sigma V^T)(V\Sigma U^T)$$

and since V is an orthogonal matrix($V^TV = I$),

$$XX^T = U\Sigma^2 U^T$$

and the correspondence is easily seen.

For each experiment, we need the original dataset and the new dataset obtained by applying the PCA. Proportion of variance is an important value in PCA which gives the main idea of how much variance this new attribute covered. Our selection uses this value to be the threshold and we choose different thresholds for selecting new subsets of data from the original one. We then obtain different datasets with threshold values of 95%, 90%, …, 50%.

5.3 10-fold cross validation

Cross validation(CV) has an alias as rotation estimation [74-76]. CV is a model evaluation method and be confirmed that is better than residuals. One important reason for this is that residual evaluations do not give an estimate for new predictions with the data it has not been seen. One possible solution to this problem is to separate the entire data set. When training a learner. Some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model. This is the basic idea for a whole class of CV.

The holdout method is the simplest kind of cross validation. The data set is separated into two sets, called the training set and the testing set. The function approximator fits a function using the training set only. Then the function approximator is asked to predict the output values for the data in the testing set (it has never seen these output values before). The errors it makes are accumulated as before to give the mean absolute test set error, which is used to evaluate the

model. The advantage of this method is that it is usually preferable to the residual method and takes no longer to compute. However, its evaluation can have a high variance. The evaluation may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made.

K-fold cross validation is one way to improve over the holdout method. The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set k-1 times. The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set k different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over.[77]

In [78], Arlot, Sylvain and Celisse, Alain mentioned that when the goal of model selection is estimation, it is often reported that the optimal K is between 5 and 10 , because the statistical performance does not increase a lot for larger values of K , and averaging over less than 10 splits remains computationally feasible[35]. We will use 10-fold cross validation for the following experiment.

Weka has the in-build CV function. For the whole process with 10-fold cross validation, see Figure 0-1. We use the arff Loader to load the data file which have been pre-processed by PCA, then pass the data set to the Cross-Validation Fold Maker. After this, we use the four classifiers for the cross validation. The result recorded by export the output from the Classifier Performance Evaluator.

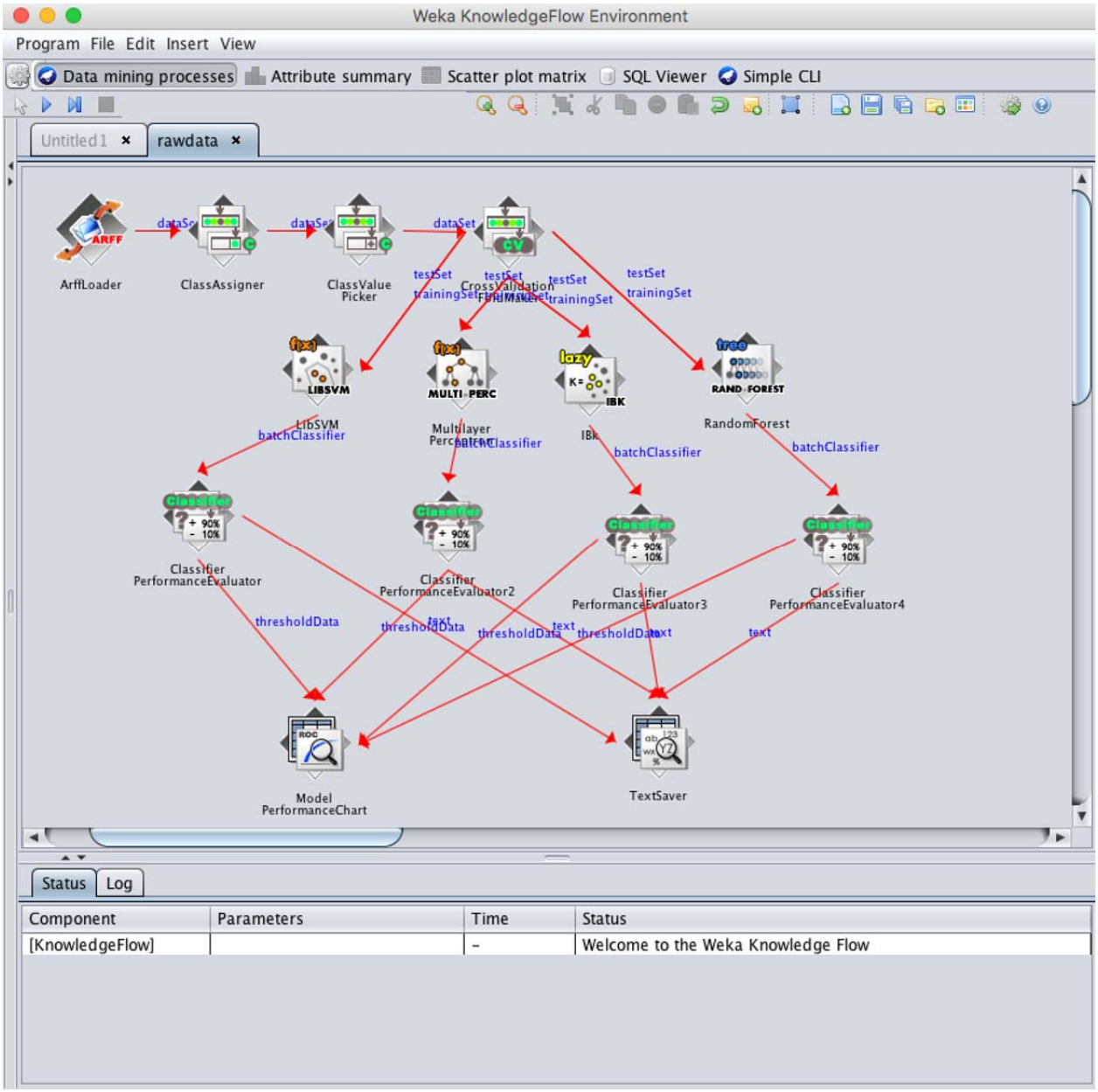

Figure 0-1: Weka KnowledgeFlow Chart for 10-fold Cross Validation

5.4 Ratio Comparison

As another way to measure the performance of PCA, we use the ratio comparison method. This method is as a follow up method of CV. We split the data set into two different data sets. One for training and the other for testing. Unlike K-fold cross validation, we split the data by vary percentages. The specific percentage we choose is 90%, 80%, 70%, 60% for the training set respectively. The whole process of the ratio comparison can be check from Figure 0-2. We use the data set split maker after we loaded the data set. The test set and train set were tested by the four classifiers.

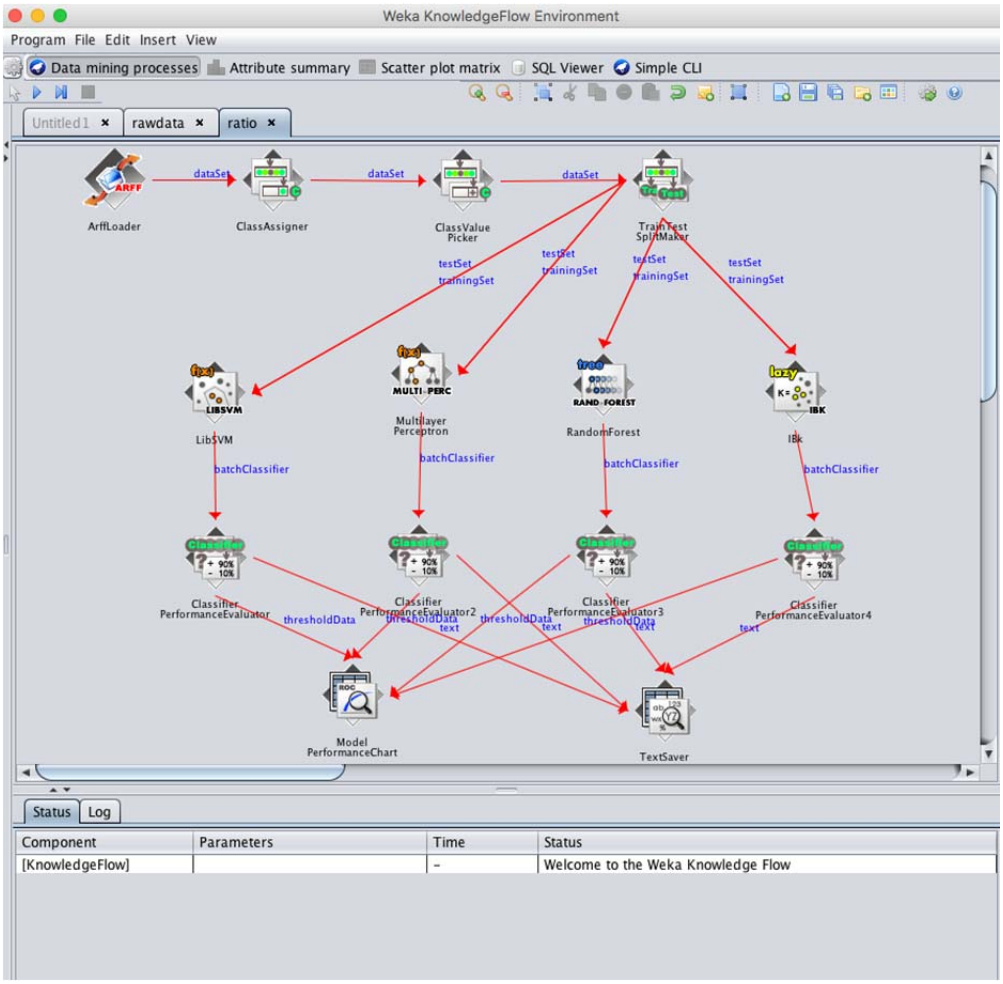

Figure 0-2:Weka KnowledgeFlow Chart for Ratio Comparison

5.5 Results and Discussion

5.5.1 Principal Component Analysis Dataset List

We applied PCA on the colon and leukemia datasets. The variance table returned by PCA is listed in Table 0-1 and Table 0-2.

Table 0-1:Colon Dataset Thresholds and Attribute Selection

Colon Dataset Thresholds	Cumulative Proportion	Attributes Selected
100%(Raw)	100%	2001
95%	95.013%	45
90%	90.520%	35
85%	85.677%	27
80%	80.006%	20
75%	75.545%	16
70%	71.429%	13

65%	66.004%	10
60%	61.154%	8
55%	57.701%	7
50%	53.180%	6

The experiment is based on the 11 datasets shown in Table 0-1 and Table 0-2. The 100% dataset threshold means we use the raw data as input for the experiments. The 95% to 50% datasets are chosen by PCA method.

Table 0-2:Leukemia Dataset Thresholds and Attributes Selection

Dataset Thresholds	Cumulative Proportion	Attributes Selected
100%(Raw)	100%	7130
95%	95.192%	59
90%	90.244%	49
85%	85.560%	41
80%	80.232%	33
75%	75.638%	27
70%	70.261%	21
65%	65.997%	17
60%	60.570%	13
55%	55.557%	10
50%	51.440%	8

5.5.2 10-fold cross validation

The results are listed in Table 3. The accuracy (correctly classified instances) is given by:

$$accuracy = \frac{TP + TN}{N}$$

where TP indicates the True Positive instances, TN indicates the True Negative instances and N is the total number of instances in the test set.

Table 5-3 shows the accuracy and Area Under ROC curve (AUC) for the four classifiers on the colon dataset. K-nearest-neighbor and Random Forest algorithms shows the highest accuracy for

a threshold of 60% by PCA. SVM algorithm shows the highest accuracy for the raw data and for 90% threshold by PCA. Neural Network algorithm shows the highest accuracy for a threshold of 90% by PCA. All the four classifiers show improvement in accuracy for some threshold value of PCA as compared to raw data except for SVM. Figure 0-3 shows the results of the 10-folds cross validation for colon dataset.

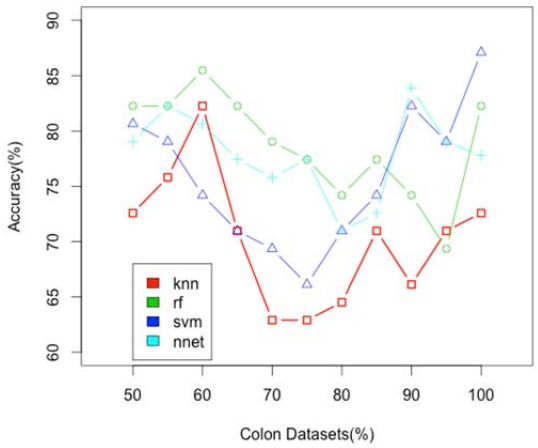

Figure 0-3:10-fold cross validation results for colon dataset

Table 5-3:10-Folds Cross Validation For Colon Dataset

Data Mining Methods	Dataset	Accuracy	ROC Area(AUC)
KNN	***Raw***	72.5806%	0.699
	95%	70.9677%	0.680
	90%	66.129%	0.648
	85%	70.9677%	0.728
	80%	64.5161%	0.619
	75%	62.9032%	0.581
	70%	62.9032%	0.592
	65%	70.9677%	0.706
	60%	82.2581%	0.815
	55%	75.8065%	0.752
	50%	72.5806%	0.744
Random Forest	***Raw***	82.2581%	0.885
	95%	69.3548%	0.879
	90%	74.1935%	0.877
	85%	77.4194%	0.840
	80%	74.1935%	0.812
	75%	77.4194%	0.845
	70%	79.0323%	0.855
	65%	82.2581%	0.873

	60%	85.4839%	0.892
	55%	82.2581%	0.881
	50%	82.2581%	0.872
SVM	***Raw***	87.0968%	0.886
	95%	79.0323%	0.868
	90%	82.2581%	0.893
	85%	74.1935%	0.805
	80%	70.9677%	0.797
	75%	66.129%	0.759
	70%	69.3548%	0.723
	65%	70.9677%	0.830
	60%	74.1935%	0.903
	55%	79.0323%	0.869
	50%	80.6452%	0.881
Neural Network	***Raw***	77.8%	0.857
	95%	79.0323%	0.851
	90%	83.871%	0.895
	85%	72.5806%	0.819
	80%	70.9677%	0.777
	75%	77.4194%	0.845
	70%	75.8065%	0.786
	65%	77.4194%	0.805
	60%	80.6452%	0.843
	55%	82.2581%	0.834
	50%	79.0323%	0.826

Table 0-4:10-FOLD CROSS VALIDATION FOR LEUKEMIA DATASET

Data Mining Methods	**Dataset**	**accuracy**	**ROC Area (auc)**
KNN	***Raw***	65.2778%	0.505
	95%	72.2222%	0.656
	90%	75%	0.667
	85%	77.7778%	0.719
	80%	81.9444%	0.811
	75%	83.3333%	0.829
	70%	93.0556%	0.911
	65%	91.6667%	0.887
	60%	88.8889%	0.861
	55%	90.2778%	0.874
	50%	91.6667%	0.874
Random Forest	***Raw***	76.3889%	0.889
	95%	79.1667%	0.918
	90%	84.7222%	0.945
	85%	84.7222%	0.952
	80%	93.0556%	0.963
	75%	93.0556%	0.958
	70%	94.4444%	0.978
	65%	94.4444%	0.974

	60%	91.6667%	0.963
	55%	90.2778%	0.968
	50%	91.6667%	0.969
SVM	***Raw***	98.6111%	0.998
	95%	97.2222%	0.995
	90%	86.1111%	0.969
	85%	87.5%	0.959
	80%	93.0556%	0.968
	75%	90.2778%	0.977
	70%	90.2778%	0.974
	65%	88.8889%	0.963
	60%	93.0556%	0.969
	55%	90.2778%	0.933
	50%	86.1111%	0.962
Neural Network	***Raw***	81.9444%	0.865
	95%	83.3333%	0.877
	90%	87.5%	0.917
	85%	90.2778%	0.934
	80%	90.2778%	0.970
	75%	90.2778%	0.977
	70%	88.8889%	0.980
	65%	88.8889%	0.971
	60%	93.0556%	0.971
	55%	91.6667%	0.951
	50%	93.0556%	0.974

Table 0-4 shows the accuracy and Area Under ROC curve (AUC) for the four classifiers on the leukemia dataset. K-nearest-neighbor shows the highest accuracy for a threshold of 70% by PCA. Random Forest shows the highest accuracy for a threshold of 65% and 70% by PCA. SVM shows the highest accuracy for the raw data and next highest accuracy for a threshold of 95% by PCA. Neural Network shows the highest accuracy for a threshold of 60% and 50% by PCA. Figure 0-4 shows the results of the 10-folds cross validation for leukemia dataset.

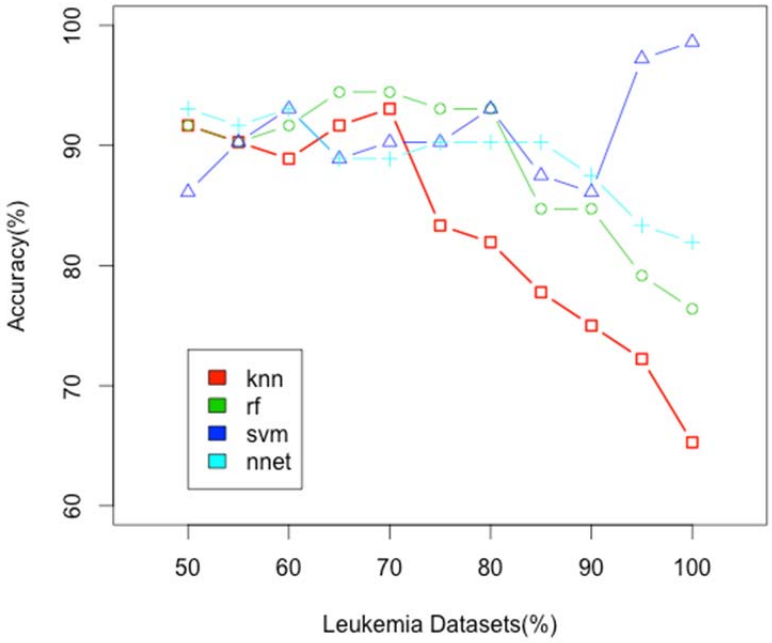

Figure 0-4:10-fold cross validation results for leukemia dataset

5.5.3 Ratio Comparison

The second method we use for this experiment is that we split the dataset to a training set and test set by different ratio in 90%:10%,80%:20%,70%:30% and 60%:40%. All the result applied to the data which preprocessed by PCA. We show the results in Table 0-5 and Table 0-6.

Figure 0-5, Figure 0-6, Figure 0-7, Figure 0-8 show the accuracy for the four algorithms for different ratios of training and test datasets. Further discussion is given below.

Table 0-5:Ratio Validation Results for Colon Dataset

		9:1		**8:2**		**7:3**		**6:4**	
	Method	***accuracy***	***auc***	***accuracy***	***auc***	***accuracy***	***auc***	***accuracy***	***auc***
Raw	***Knn***	83.33%	0.75	83.33%	0.75	78.95%	0.683	76%	0.7
	Svm	100%	1	91.667%	1	89.47%	0.9	88%	0.93
	Rf	100%	1	100%	1	89.47%	0.892	84%	0.9
	Nnet	100%	0.85	100%	0.916	89.47%	0.935	88%	0.893
95%	***Knn***	83.33%	0.875	83.3%	0.875	78.95%	0.867	76%	0.775
	Svm	100%	1	91.67%	1	89.47%	0.917	80%	0.91
	Rf	66.7%	0.875	66.67%	0.797	68.42%	0.725	68%	0.77
	Nnet	100%	1	83.3%	0.875	73.68%	0.800	76%	0.81
90%	***Knn***	83.33%	0.875	66.67%	0.688	57.89%	0.642	56%	0.575
	Svm	100%	1	100%	1	89.47%	0.95	84%	0.92
	Rf	66.67%	1	75%	0.734	73.68%	0.833	76%	0.845
	Nnet	100%	1	100%	1	94.74%	0.983	88%	0.91
85%	***Knn***	66.67%	0.625	58.33%	0.563	63.16%	0.675	56%	0.575
	Svm	100%	1	91.67%	1	84.21%	0.95	80%	0.910
	Rf	83.33%	1	83.33%	0.875	63.16%	0.783	68%	0.82
	Nnet	100%	1	91.67%	1	84.21%	0.967	88%	0.88
80%	***Knn***	66.67%	0.5	58.33%	0.438	63.16%	0.4	60%	0.45
	Svm	83.33%	1	91.67%	0.938	84.21%	0.850	76%	0.84
	Rf	83.33%	1	83.33%	0.969	68.42%	0.817	64%	0.83
	Nnet	100%	1	83.33%	0.938	73.68%	0.833	72%	0.77
75%	***Knn***	66.67%	0.5	58.33%	0.438	63.16%	0.4	56%	0.35
	Svm	66.67%	0.75	91.67%	1	78.95%	0.842	72%	0.87
	Rf	83.33%	1	91.67%	1	84.21%	0.833	68%	0.8
	Nnet	100%	1	66.67%	0.906	78.95%	0.817	72%	0.78
70%	***Knn***	66.67%	0.5	58.33%	0.438	63.16%	0.4	60%	0.375
	Svm	100%	1	66.67%	0.875	73.68%	0.875	64%	0.835
	Rf	83.33%	0.875	66.67%	0.906	78.95%	0.85	72%	0.94
	Nnet	66.67%	1	66.67%	0.906	78.95%	0.833	84%	0.92
65%	***Knn***	66.67%	0.5	75%	0.625	73.68%	0.558	68%	0.575
	Svm	100%	1	75%	0.938	78.95%	0.858	64%	0.81
	Rf	83.33%	1	83.33%	0.938	73.68%	0.933	64%	0.835
	Nnet	100%	1	83.33%	1	84.21%	0.917	72%	0.92
60%	***Knn***	83.33%	0.75	83.33%	0.75	84.21%	0.808	84%	0.825
	Svm	100%	1	100%	1	78.95%	0.933	84%	0.95
	Rf	100%	1	83.33%	1	78.95%	0.942	64%	0.955
	Nnet	100%	1	100%	1	84.21%	0.933	84%	0.98
55%	***Knn***	66.67%	0.625	75%	0.688	78.95%	0.775	80%	0.8
	Svm	83.33%	1	91.67%	1	63.16%	0.958	88%	0.95
	Rf	100%	1	91.67%	1	84.21%	0.917	56%	0.96
	Nnet	100%	1	100%	1	78.94%	0.9	80%	0.94
50%	***Knn***	66.67%	0.625	75%	0.688	73.68%	0.65	72%	0.675
	Svm	100%	1	100%	1	89.47%	0.9	80%	0.88
	Rf	100%	1	100%	1	73.68%	0.908	60%	0.95
	Nnet	100%	1	83.33%	1	89.47%	0.883	80%	0.92

Table 0-6:Ratio Validation Results for Leukemia Dataset

		9:1		8:2		7:3		6:4	
	Method	***accuracy***	***auc***	***accuracy***	***auc***	***accuracy***	***auc***	***accuracy***	***auc***
Raw	***Knn***	28.5714%	0.583	35.7143%	0.550	54.5455%	0.545	58.6207%	0.546
	Svm	100%	1	100%	1	100%	1	100%	1
	Rf	14.2857%	1	28.5714%	1	59.0909%	0.888	55.1724%	0.798
	Nnet	71.4286%	1	42.8571%	0.9	63.6364%	0.983	68.9655%	0.870
95%	***Knn***	71.4286%	0.833	78.5714%	0.775	81.8182%	0.818	72.4138%	0.714
	Svm	100%	1	100%	1	95.4545%	1	96.5517%	1
	Rf	57.1429%	1	35.7143%	0.975	59.0909%	0.921	65.5172%	0.825
	Nnet	71.4286%	1	42.8571%	0.825	72.7273%	0.818	72.4138%	0.731
90%	***Knn***	42.8571%	0.667	50%	0.650	54.5455%	0.545	58.6207%	0.538
	Svm	85.7143%	1	85.7143%	1	90.9091%	0.975	89.6552%	0.976
	Rf	42.8571%	1	50%	1	59.0909%	0.893	65.5172%	0.873
	Nnet	71.4286%	1	64.2857%	0.9	68.1818%	0.851	72.4138%	0.793
85%	***Knn***	42.8571%	0.667	50%	0.650	54.5455%	0.545	62.069%	0.584
	Svm	71.4286%	1	71.4286%	0.975	86.3636%	0.975	89.6552%	0.976
	Rf	85.7143%	1	57.1429%	1	63.6364	1	68.9655%	0.962
	Nnet	57.1429%	1	71.4286%	0.925	72.7273%	0.901	79.3103%	0.870
80%	***Knn***	57.1429%	0.750	57.1429%	0.7	68.1818%	0.682	72.4138%	0.7
	Svm	71.4286%	1	71.4286%	1	90.9091%	0.942	89.6552%	0.962
	Rf	85.7143%	1	71.4286%	1	81.8182%	0.992	72.4138%	0.988
	Nnet	57.1429%	1	85.7143%	1	90.9091%	0.934	72.4138%	0.861
75%	***Knn***	85.7143%	0.917	92.8571%	0.95	81.8182%	0.818	75.8621%	0.752
	Svm	71.4286%	1	85.7143%	0.975	86.3636%	0.934	89.6552%	0.942
	Rf	100%	1	64.2857%	1	77.2727%	0.996	79.3103%	0.981
	Nnet	71.4286%	1	85.7143%	0.975	81.8182%	0.967	79.3103%	0.875
70%	***Knn***	100%	1	100%	1	95.4545%	0.955	93.1034%	0.923
	Svm	100%	1	85.7143%	1	100%	1	96.5517%	1
	Rf	100%	1	78.5714%	1	95.4545%	1	89.6552%	0.955
	Nnet	100%	1	85.7143%	1	90.9091%	1	89.6552%	0.976
65%	***Knn***	85.7143%	0.917	92.8571%	0.95	95.4545%	0.955	89.6552%	0.892
	Svm	85.7143%	1	85.7143%	1	95.4545%	1	96.5517%	1
	Rf	100%	1	92.8571%	1	95.4545%	1	96.5517%	1
	Nnet	85.7143%	1	100%	1	100%	1	100%	1
60%	***Knn***	85.7143%	0.917	85.7143%	0.9	90.9091%	0.909	89.6552%	0.892
	Svm	100%	1	78.5714%	1	90.9091%	1	93.1034%	1
	Rf	100%	1	92.8571%	1	95.4545%	1	89.6552%	1
	Nnet	100%	1	100%	1	100%	1	100%	1
55%	***Knn***	100%	1	71.4286%	0.8	77.2727%	0.773	82.7586%	0.815
	Svm	100%	1	92.8571%	0.975	95.4545%	0.992	96.5517%	0.990
	Rf	100%	1	92.8571%	1	95.4545%	1	93.1034%	1
	Nnet	100%	1	100%	1	95.4545%	1	100%	1
50%	***Knn***	100%	1	78.5714%	0.850	81.8182%	0.818	86.2069%	0.853
	Svm	85.7143%	1	92.8571%	1	95.4545%	1	96.5517%	1
	Rf	100%	1	92.8571%	1	95.4545%	0.996	96.5517%	1
	Nnet	100%	1	100%	1	100%	1	100%	1

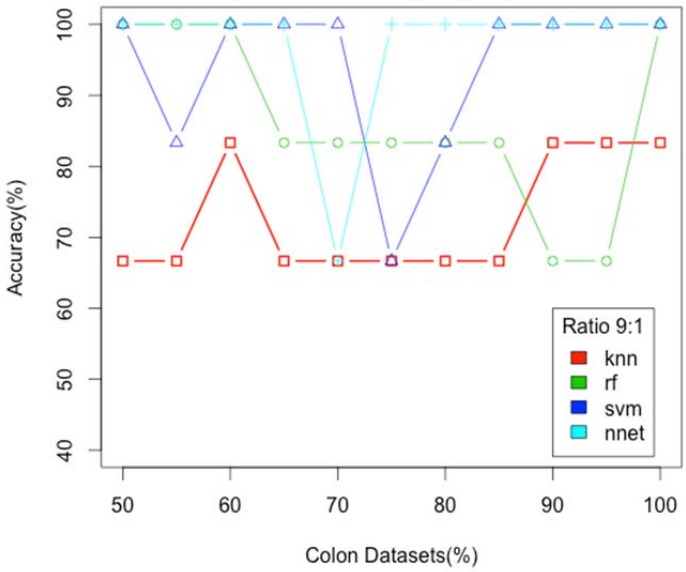

Figure 0-5:Accuracy comparison in Ratio 9:1

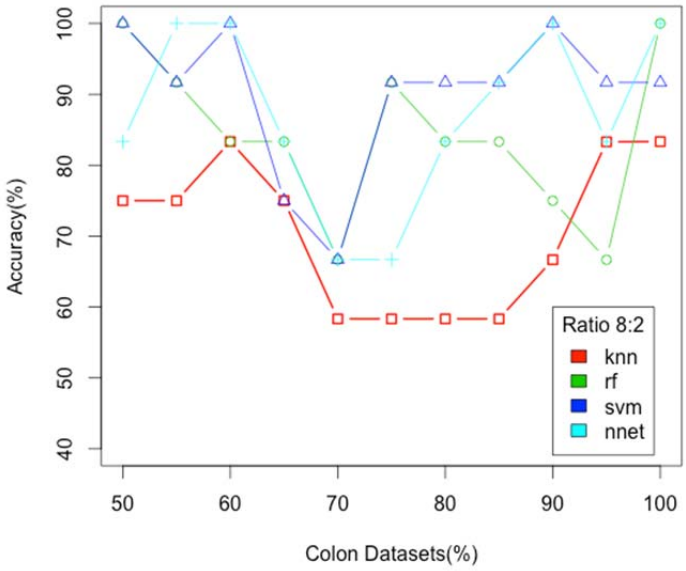

Figure 0-6:Accuracy comparison in Ratio 8:2

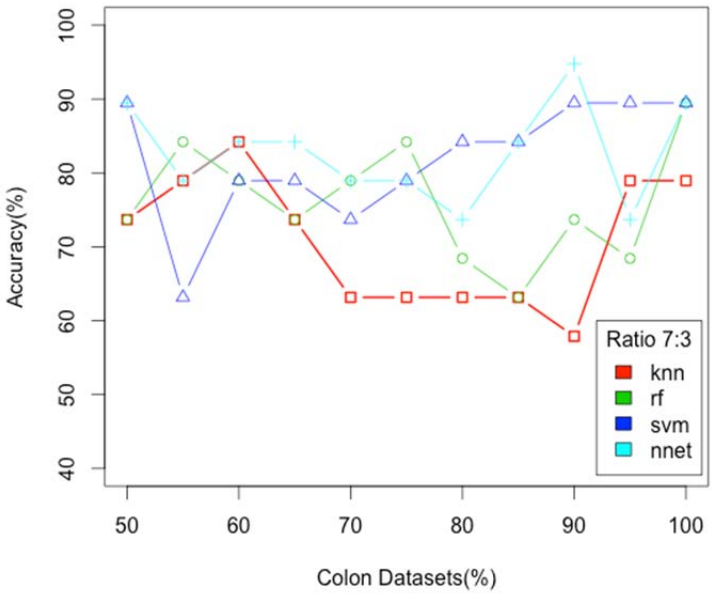

Figure 0-7:Accuracy comparison in Ratio 7:3

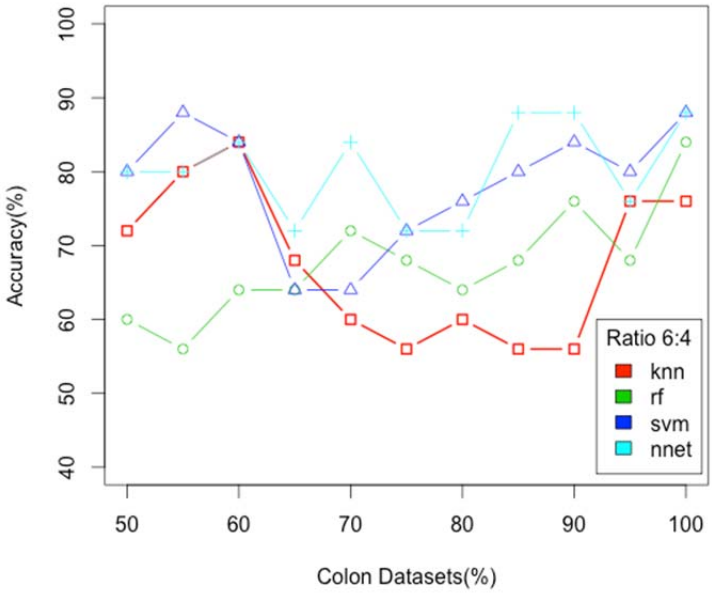

Figure 0-8:Accuracy comparison in Ratio 6:4

5.5.4 Discussion and Conclusion

In Table 5-3 for the colon dataset, we observe that K-nearest-neighbor algorithm gives the highest accuracy of 82.3% for a threshold of 60% by PCA as compared to 72.6% for the raw data, an increase of 9.7% in accuracy by applying PCA. Random Forest algorithm gives the highest accuracy of 85.5% for a threshold of 60% by PCA as compared to 82.3% for the raw data, an increase of 3.3% in accuracy. Neural Network algorithm gives the highest accuracy of 83.9% for a threshold of 90% by PCA as compared to 77.8% for the raw data, an increase of 6.1% in accuracy. However, in SVM, the highest accuracy was observed as 87% for the raw data and 82% accuracy for a threshold of 90% by PCA, a decrease of 5% in accuracy.

In Table 0-4 for the leukemia dataset, we observe that K-nearest-neighbor algorithm gives the highest accuracy of 93% for a threshold of 70% by PCA as compared to 65% for the raw data, an increase of 28% in accuracy by applying PCA. Random Forest algorithm gives the highest accuracy of 94.4% for a threshold of 65% and 70% by PCA as compared to 76.4% for the raw data, an increase of 18% in accuracy. Neural Network algorithm gives the highest accuracy of 93% for a threshold of 50% and 60% by PCA as compared to 81.9% for the raw data, an increase of 11% in accuracy. However, in SVM, the highest accuracy was observed as 98.6% for the raw data as compared to 97.2% for a threshold of 95% by PCA, a decrease of 3.6% in accuracy.

SVM was tested for four different kernels – linear, polynomial, radial basis function and sigmoid function. The linear kernel gave the best results. For the exception of SVM, all other algorithms increased the accuracy of classification by applying PCA. Greater increase in accuracy was observed in leukemia dataset than the colon dataset. Figure 0-3 and Figure 0-4 show the results for 10-fold cross validation for the colon and leukemia datasets, respectively.

Table 0-5 and Table 0-6 show the accuracy and AUC of the colon and leukemia datasets respectively for raw data and different thresholds of PCA and by taking different ratios of training and test data.

In the colon dataset, we observe that highest accuracy is achieved for the training to test ratio of 9:1 and second highest accuracy for the ratio 8:2 for all the algorithms with and without using PCA.

In training to test ratio 9:1, Random Forest, SVM and Neural Network algorithms give an accuracy of 100% whereas k-nearest-neighbor gives an accuracy of 83.3% for the raw data. PCA maintains the accuracy of 100% at the threshold of 50% and 60% for Random Forest, SVM and Neural Network and maintains the accuracy of 83.3% at the threshold of 60%, 90% and 95% for the k-nearest-neighbor algorithm. Figure 0-5 shows the comparison of accuracy for the four algorithms for the ratio 9:1.

In training to test ratio 8:2, Random Forest and Neural Network algorithms give an accuracy of 100%, SVM has an accuracy of 91.6% and KNN has an accuracy of 83.3% for the raw data. PCA maintains the accuracy of 100% at thresholds of 55% and 60% for Neural Network and at the threshold of 50% for Random Forest. PCA increased the accuracy of SVM from 91.6% to 100% at the thresholds of 50%, 60% and 90%. PCA maintains the accuracy of KNN at 83.3% at a threshold of 60%. Figure 0-6 shows the comparison of accuracy for the four algorithms for the ratio 8:2.

In training to test ratio 7:3, PCA increases the accuracy of KNN from 78.95% to 84.21% at a threshold of 60% and increases the accuracy of Neural Network from 89.5% to 94.7% at a threshold of 90%. PCA maintains the accuracy of SVM at 89.5% at the thresholds of 50%, 90%

and 95%. However, the accuracy of Random Forest is decreased from 89.5% to 84.2% at thresholds of 55% and 80%. Figure 0-7 shows the comparison of accuracy for the four algorithms for the ratio 7:3.

In training to test ratio of 6:4, PCA increases the accuracy of KNN from 76% to 84% at a threshold of 60%. PCA maintains the accuracy of SVM and Neural Network at 88% at a threshold of 55% for SVM and at 85% and 90% for Neural Network. However, the accuracy of Random Forest decreased from 88% to 76% at a threshold of 95%. Figure 0-8 shows the comparison of accuracy for the four algorithms for the ratio 6:4.

Overall, PCA either maintains the accuracy of all the four algorithms or increases the accuracy except for Random Forest at ratios of 7:3 and 6:4.

For the leukemia dataset experiment, we observe from Table 0-6 that the highest accuracy is achieved for the training to test ratio of 9:1.

In training to test ratio of 9:1, PCA increased the accuracy of KNN from 28.6% to 100% at the thresholds of 50%, 55% and 70%, an increase of 71.4% in accuracy. PCA increased the accuracy of Random Forest from 14.3% to 100% at threshold of 50%, 55%, 60%, 65%, 70%, 75%, an increase of 85.7% in accuracy. PCA increased the accuracy of Neural Network from 71.4% to 100% at threshold of 50%, 55% and 70%, an increase of 28.6% in accuracy. PCA maintains the accuracy of SVM at 100% at a threshold of 50%, 60%, 65%, 70%, 85%, 90% and 95%.

In training to test ratio of 8:2, PCA increased the accuracy of KNN from 35.7% to 100% at a threshold of 70%, an increase of 64.3% in accuracy. PCA increased the accuracy of Random Forest from 28.6% to 92.9% at threshold of 50%, 55%, 60% and 65%, an increase of 64.3% in

accuracy. PCA increased the accuracy of Neural Network from 42.9% to 100% at thresholds of 50%, 55%, 60%, and 65%, an increase of 57% in accuracy. PCA maintains the accuracy of 100% for SVM at a threshold of 95%.

In training to test ratio of 7:3, PCA increased the accuracy of KNN from 54.5% to 95.5% at threshold of 65% and 70%, an increase of 41%. PCA increased the accuracy of Random Forest from 59% to 95.5% at thresholds of 50%, 55%, 60%, 65%, and 70%, an increase of 36.5% in accuracy. PCA increased the accuracy of Neural Network from 63.6% to 100% at thresholds of 50%, 60%, and 65%, an increase of 36.4% in accuracy. PCA maintains the accuracy of SVM at 100% at a threshold of 70%.

In training to test ratio of 6:4, PCA increased the accuracy of KNN from 58.6% to 93% at a threshold of 70%, an increase of 34.4% in accuracy. PCA increased the accuracy of Random Forest from 55% to 96.5% at thresholds of 50% and 65%, an increase of 41.5% in accuracy. PCA increased the accuracy of Neural Network from 68.9% to 100% at thresholds of 50%, 55%, 60% and 65%, an increase of 31% in accuracy. However, the accuracy of SVM decreased from 100% to 96.5% at thresholds of 50%, 55%, 70% and 95%.

From the two datasets that PCA increases the accuracy of the four classifiers at different thresholds. There are significant improvements in the accuracy for leukemia dataset.

In this chapter, we applied the Principle Component Analysis (PCA) on colon dataset and the leukemia dataset and we compared the accuracy for four different classifiers. Support Vector Machine and Neural Network gave the best performance among the four methods. The experiments included 10-fold cross validation and different training to test ratios of 9:1, 8:2, 7:3 and 6:4.

PCA increased the accuracy of the four classifiers for the colon and leukemia datasets. However, it was observed that there were significant improvements in the performance of most of the classifiers with 10-folds cross validation. The improvements were more significant for the leukemia dataset. In the case of different training to test ratios, PCA maintained the accuracy of the classifiers or increased the accuracy for the colon dataset. However, PCA increased the accuracy of the classifiers significantly for the leukemia dataset.

PCA was selected as a feature selection method to test for increase in accuracy of classifiers on test datasets. The results were promising and it gives us further incentive to test the accuracy of the classifiers with other feature selection algorithms in future.

Chapter 6

Improved Microarray Data Analysis using Feature Selection Methods with Machine Learning Methods

The gene expression profiling techniques by DNA microarrays provide the analysis of large amount of genes [71]. The number of gene expression data of microarray has grown exponentially. Most of the gene expression dataset has hundreds of variables leading to data with very high dimensionality. This makes the job for classification more difficult. It is of great importance to find the key gene expressions which can best describe the phenotypic trait [72]. Feature selection methods become a possible solution for this problem. It gives us a way to reduce computing time as well as improving the performance of prediction and also provides us a better chance to understand the data in machine learning tasks [79].

In this study, we use colon cancer dataset and leukemia cancer dataset for the experiment. We use discretization method combined with some well-known feature selection methods for the task of preprocessing. After that, we apply 10-folds cross validation with KNN, SVM, RF and NN classifiers on the dataset. The performance of the classifiers is based on accuracy and the AUC (area under the curve) values.

The results demonstrate that mRMR is the best method for improving the performance of microarray data analysis and discretization method can greatly improve the performance of

mRMR method and MaxRel method. Also, the results give the best numbers for specific feature selection method and classifiers.

In this chapter, we present the state of our experimental environment in Section 6.1.The Process of this experiment is explained in Section6.2. The discussion and analysis of the results are presented in Section 6.3. Section 6.4 presents the conclusions of this study and list some future perspective.

6.1 Tools in this Experiment

In this experiment, Weka is the main testing tools for 10-folds cross validation. The version we use in this experiment is 3-7-13-oracle-jvm and can be download on its official website. The platform we use in this experiment is Mac OS 10.11.1.

The Weka KnowledgeFlow Environment presents a "data-flow" inspired interface to Weka. The user can select Weka components from a tool bar, place them on a layout canvas and connect them together in order to form a "knowledge flow" for processing and analyzing data. At present, all of Weka's classifiers and filters are available in the KnowledgeFlow along with some extra tools.

We use Weka KnowledgeFlow for the 10-folds cross validation of different feature selection methods. The flow chart is showed in Figure 0-1.

6.2 Process of Experiment

The first step for our experiment is to generate the subsets of the data set. For each feature selection method, we run the algorithms on different platforms and different programs in order to

generate subsets. Most of the result are given as an attribute list ranked by score. An example listed in Figure 0-1. The columns represent to rank number, attribute number, attribute name, score respectively.

1	**14**	**gene14**	**0.349**
2	**377**	**gene377**	**0.331**
3	**493**	**gene493**	**0.321**
4	**1917**	**gene1917**	**0.306**
5	**765**	**gene765**	**0.295**
6	**249**	**gene249**	**0.273**
7	**625**	**gene625**	**0.273**
8	**1771**	**gene1771**	**0.271**
	…		

Figure 0-1: Format of an output results

In our experiment, we choose 5,10,20,30,50,100,200 features, which means 5,10, 20, …, etc., features with higher scores are selected after we applied the feature selection methods. We use python to transform and gather subset from origin data set. Core code for this transformation is given in Appendix B: Codes (See Table 0-2 and Table 0-3). After we get all the subsets, we then apply 10-folds cross validation with SVM, KNN, NN and RF classification algorithms using Weka knowledge workflow.

6.3 Discussion and Analysis

Results are listed in Table 0-1 ~ Table 0-4, as we can see from Table 0-1, we record all the result by accuracy indicator. PLS shows the best results for all the four classifiers. It shows 0 error rate in SVM no matter how many features were selected. Next to PLS, the mRDD method gets a best performance in RF classifier. It reaches 95.1613 which is the same as PLS + RF.

Not all the methods can reach a good performance such as higher than 90. With KNN and SVM and NN classifier, mRDD, mRDQ and PLS can perform higher than 90%. With RF classifier, all method can get higher than 90% except for MRC as well as QPFS. For KNN classifier, most

good results get when 20 or 30 features selected. For SVM and NN, most good results get when 5, 10 feature selected. For RF, most good results get when 5, 50 feature selected.

Table 0-2 shows the AUC part of the colon dataset. We notice that PLS still shows really good performance except for RF, and mRDD shows the best results. If we use 0.9 as a threshold for a "good result", then we will find for KNN, mRDD, mRDQ, MRD and PLS get good results. For SVM, mRCQ, mRDD, mRDQ, MRC, MRD, QPFS and PLS get good results. For NN and RF, all the methods get good results.

By compare which number of features selected within certain feature selection method, we can see that most good result get when 5, 10 or 30 features selected.

Unlike Colon data set. Leukemia data set gave us different results in Table 0-3 and Table 0-4. PLS and other discretized methods all reach the peak of 100% accuracy. The discretized methods include mRDD, mRDQ, MRD. When we compare that in accuracy, we can see that PLS+SVM and RF + mRDD get 100% accuracy all the time. For AUC, mRDD gets 1.0 AUC value all the time with SVM, NN, RF. If we compare the good results among all the methods and all the classifiers, we can see that QPFS gets all the worse "good result". This means that on Leukemia data set, QPFS is not enough consistent and not enough strong for a feature selection method.

mRDD, mRDQ, MRD, PLS reach the highest performance on 0 error rate when we select accuracy as the indicator.

mRDD, mRDQ, MRD, PLS reach the highest performance on 1.0 AUC value when we select AUC as the indicator.

Table 0-1:Colon-Accuracy Results

KNN	**5**	**10**	**20**	**30**	**50**	**100**	**200**
FS_Methods							
mRCD	85.48	79.03	79.03	**88.71**	85.48	87.10	83.87
mRCQ	83.87	85.48	82.26	82.26	**88.71**	87.10	83.87
mRDD	91.94	93.55	91.94	**95.16**	91.94	87.10	85.48
mRDQ	88.71	88.71	**93.55**	**93.55**	90.32	87.10	85.48
MRC	87.10	83.87	87.10	82.26	85.48	**88.71**	85.48
MRD	85.48	87.10	**88.71**	87.10	83.87	83.87	80.65
QPFS	82.26	82.26	**85.48**	**85.48**	82.26	**85.48**	83.87
PLS	**100.00**	98.39	80.65	79.03	74.19	75.81	72.58
SVM							
mRCD	**88.71**	79.03	80.65	80.65	82.26	83.87	85.48
mRCQ	85.48	83.87	**87.10**	85.48	83.87	83.87	85.48
mRDD	88.71	**91.94**	87.10	87.10	87.10	85.48	85.48
mRDQ	87.10	80.65	83.87	85.48	**90.32**	**90.32**	87.10
MRC	**88.71**	80.65	80.65	87.10	79.03	82.26	87.10
MRD	82.26	**85.48**	80.65	80.65	80.65	83.87	**85.48**
QPFS	**88.71**	**88.71**	82.26	87.10	82.26	83.87	85.48
PLS	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
NN							
mRCD	85.48	82.26	80.65	83.87	83.87	85.48	**87.10**
mRCQ	**87.10**	82.26	85.48	83.87	85.48	85.48	**87.10**
mRDD	82.26	**91.94**	87.10	87.10	90.32	87.10	87.10
mRDQ	87.10	85.48	**90.32**	**90.32**	**90.32**	87.10	87.10
MRC	**87.10**	79.03	83.87	85.48	83.87	83.87	**87.10**
MRD	83.87	82.26	83.87	83.87	82.26	**85.48**	82.26
QPFS	**88.71**	80.65	77.42	83.87	87.10	83.87	85.48
PLS	**100.00**	**100.00**	**100.00**	**100.00**	93.55	95.16	96.77
RF							
mRCD	85.48	85.48	**90.32**	87.10	**90.32**	87.10	85.48
mRCQ	82.26	88.71	87.10	87.10	88.71	**90.32**	88.71
mRDD	**95.16**	91.94	91.94	88.71	91.94	91.94	88.71
mRDQ	**91.94**	**91.94**	90.32	90.32	**91.94**	88.71	88.71
MRC	87.10	85.48	87.10	87.10	**88.71**	87.10	87.10
MRD	88.71	85.48	**90.32**	**90.32**	88.71	88.71	85.48
QPFS	**88.71**	87.10	85.48	87.10	**88.71**	**88.71**	**88.71**
PLS	93.55	**95.16**	91.94	83.87	70.97	83.87	77.42

Table 0-2:Colon-AUC results

KNN	**5**	**10**	**20**	**30**	**50**	**100**	**200**
FS_Methods							
mRCD	0.81	0.74	0.70	**0.84**	0.82	0.83	0.81
mRCQ	0.84	0.86	0.78	0.79	**0.87**	0.83	0.81
mRDD	**0.97**	0.97	0.95	0.95	0.96	0.89	0.82
mRDQ	0.94	0.93	**0.94**	0.94	0.92	0.87	0.83
MRC	0.85	0.77	0.82	0.78	0.79	**0.87**	0.83
MRD	0.85	0.87	**0.93**	0.91	0.85	0.84	0.83
QPFS	0.80	0.80	**0.82**	0.79	0.75	0.81	0.82
PLS	**1.00**	1.00	0.79	0.81	0.67	0.69	0.65
SVM							
mRCD	0.89	0.85	0.89	0.83	**0.90**	0.87	0.89
mRCQ	0.88	0.88	0.91	**0.92**	0.87	0.87	0.90
mRDD	0.94	**0.95**	0.92	0.92	0.92	0.93	0.93
mRDQ	0.92	0.86	0.93	0.95	0.95	**0.96**	0.94
MRC	0.90	0.87	0.85	0.90	0.85	0.88	**0.93**
MRD	0.90	**0.93**	0.92	0.84	0.86	0.89	0.90
QPFS	**0.95**	0.93	0.88	0.91	0.90	0.89	0.92
PLS	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
NN							
mRCD	0.89	0.86	0.86	0.85	**0.91**	0.89	0.90
mRCQ	0.84	**0.93**	0.92	0.90	0.90	0.89	0.90
mRDD	0.93	**0.94**	0.94	0.94	0.94	0.93	0.93
mRDQ	0.92	0.88	0.94	**0.96**	0.94	0.95	0.94
MRC	0.90	0.80	0.90	0.90	0.85	0.88	**0.92**
MRD	0.90	0.90	**0.92**	0.87	0.88	0.90	0.90
QPFS	**0.94**	0.92	0.82	0.90	0.91	0.88	0.91
PLS	**1.00**	**1.00**	**1.00**	**1.00**	0.97	0.99	1.00
RF							
mRCD	0.92	0.91	0.91	0.92	**0.93**	0.93	0.93
mRCQ	0.92	0.93	0.91	0.93	0.93	**0.94**	0.93
mRDD	**1.00**	0.98	0.98	0.98	0.98	0.97	0.94
mRDQ	0.98	**0.99**	0.97	0.98	**0.99**	0.97	0.96
MRC	**0.94**	0.92	0.90	0.90	0.92	0.92	0.93
MRD	0.91	0.93	0.94	**0.95**	0.94	0.94	0.94
QPFS	**0.94**	0.93	0.93	0.94	0.92	0.93	0.91
PLS	**0.99**	0.99	0.99	0.96	0.92	0.84	0.85

Table 0-3:Leukemia-Accuracy Results

KNN	**5**	**10**	**20**	**30**	**50**	**100**	**200**
FS_Methods							
mRCD	94.44	95.83	97.22	95.83	94.44	97.22	**98.61**
mRCQ	94.44	**98.61**	97.22	97.22	**98.61**	95.83	**98.61**
mRDD	**100.00**	98.61	97.22	**100.00**	98.61	98.61	97.22
mRDQ	97.22	98.61	**100.00**	**100.00**	**100.00**	**100.00**	98.61
MRC	94.44	90.28	**97.22**	94.44	95.83	95.83	**97.22**
MRD	97.22	**98.61**	95.83	95.83	95.83	97.22	97.22
QPFS	72.22	84.72	**87.50**	86.11	86.11	83.33	79.17
PLS	**98.61**	93.06	86.11	81.94	58.33	47.22	51.39
SVM							
mRCD	93.06	93.06	95.83	95.83	**98.61**	**98.61**	**98.61**
mRCQ	93.06	93.06	94.44	97.22	**98.61**	**98.61**	**98.61**
mRDD	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	98.61	98.61
mRDQ	98.61	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	98.61
MRC	94.44	93.06	95.83	94.44	**98.61**	**98.61**	**98.61**
MRD	97.22	95.83	98.61	94.44	**100.00**	98.61	98.61
QPFS	75.00	84.72	87.50	91.67	91.67	91.67	**94.44**
PLS	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
NN							
mRCD	90.28	93.06	97.22	95.83	**98.61**	**98.61**	**98.61**
mRCQ	91.67	93.06	97.22	**98.61**	**98.61**	**98.61**	**98.61**
mRDD	98.61	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
mRDQ	97.22	98.61	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
MRC	95.83	93.06	97.22	97.22	97.22	97.22	**98.61**
MRD	95.83	97.22	95.83	95.83	**98.61**	97.22	97.22
QPFS	73.61	91.67	88.89	88.89	94.44	90.28	**95.83**
PLS	**100.00**	**100.00**	**100.00**	**100.00**	97.22	94.44	93.06
RF							
mRCD	94.44	91.67	93.06	94.44	95.83	97.22	**98.61**
mRCQ	90.28	94.44	93.06	93.06	95.83	**97.22**	**97.22**
mRDD	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
mRDQ	93.06	98.61	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
MRC	88.89	91.67	95.83	95.83	**97.22**	**97.22**	**97.22**
MRD	97.22	**100.00**	98.61	**100.00**	**100.00**	**100.00**	98.61
QPFS	75.00	93.06	**95.83**	93.06	**95.83**	93.06	91.67
PLS	**98.61**	**98.61**	**98.61**	**98.61**	91.67	86.11	76.39

Table 0-4:Leukemia-AUC results

KNN	**5**	**10**	**20**	**30**	**50**	**100**	**200**
FS_Methods							
mRCD	0.93	0.94	0.96	0.93	0.90	0.95	**0.98**
mRCQ	0.93	**0.98**	0.96	0.95	**0.98**	0.95	**0.98**
mRDD	**1.00**	0.99	0.99	**1.00**	0.99	0.98	0.96
mRDQ	0.98	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.99
MRC	0.93	0.87	**0.96**	0.91	0.93	0.93	0.95
MRD	**1.00**	0.99	0.97	0.98	0.97	0.97	0.96
QPFS	0.72	0.83	0.87	0.85	**0.87**	0.78	0.75
PLS	**1.00**	0.95	0.85	0.78	0.62	0.53	0.53
SVM							
mRCD	0.98	0.98	1.00	1.00	**1.00**	**1.00**	**1.00**
mRCQ	0.98	0.97	0.99	1.00	**1.00**	1.00	1.00
mRDD	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
mRDQ	1.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
MRC	0.98	0.98	0.99	1.00	**1.00**	**1.00**	**1.00**
MRD	1.00	1.00	1.00	1.00	**1.00**	1.00	1.00
QPFS	0.85	0.95	0.92	0.97	**0.99**	0.98	0.99
PLS	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
NN							
mRCD	0.97	0.98	1.00	**1.00**	1.00	1.00	1.00
mRCQ	0.99	0.98	1.00	**1.00**	1.00	1.00	1.00
mRDD	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
mRDQ	1.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
MRC	0.99	0.97	**1.00**	**1.00**	1.00	1.00	1.00
MRD	1.00	**1.00**	1.00	**1.00**	**1.00**	1.00	1.00
QPFS	0.82	0.95	0.95	0.98	**1.00**	0.99	0.98
PLS	**1.00**	**1.00**	**1.00**	**1.00**	1.00	0.99	0.99
RF							
mRCD	0.99	0.99	0.99	0.99	**1.00**	**1.00**	**1.00**
mRCQ	0.98	0.98	0.99	0.99	1.00	1.00	**1.00**
mRDD	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
mRDQ	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
MRC	0.97	0.98	0.99	0.99	1.00	1.00	**1.00**
MRD	1.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.00
QPFS	0.83	0.97	0.97	0.96	0.98	0.97	**0.98**
PLS	1.00	**1.00**	**1.00**	**1.00**	**1.00**	0.95	0.89

Table 0-5:Summary of Best Accuracy

Classifier	FS Method	Colon Dataset		Leukemia Dataset	
		Best Accuracy	**# of Features**	**Best Accuracy**	**# of Features**
KNN	mRCD	88.71	30	98.61	200
	mRCQ	88.71	50	98.61	10,50, 200
	mRDD	95.16	30	100	5,30
	mRDQ	93.55	20,30	100	20-100
	MRC	88.71	100	97.22	20,200
	MRD	88.71	20	98.61	10
	QPFS	85.48	20,30,100	87.5	20
	PLS	100	5	98.61	5
SVM	mRCD	88.71	5	98.61	50-200
	mRCQ	87.1	20	98.61	50-200
	mRDD	91.94	10	100	5-50
	mRDQ	90.32	50,100	100	10-100
	MRC	88.71	5	98.61	50-200
	MRD	85.48	10,200	100	50
	QPFS	88.71	5,10	94.44	200
	PLS	100	ALL	100	ALL
NN	mRCD	87.1	200	98.61	50-200
	mRCQ	87.1	5,200	98.61	30-200
	mRDD	91.94	10	100	10-200
	mRDQ	90.32	20-50	100	20-200
	MRC	87.1	5,200	98.61	200
	MRD	85.48	100	98.61	50
	QPFS	88.71	5	95.83	200
	PLS	100	5-30	100	5-30
RF	mRCD	90.32	20,50	98.61	200
	mRCQ	90.32	100	97.22	100, 200
	mRDD	95.16	5	100	ALL
	mRDQ	91.94	5,10,50	100	20-200
	MRC	88.71	50	97.22	50-200
	MRD	90.32	20,30	100	10,30-100
	QPFS	88.71	5,50-200	95.83	20,50
	PLS	95.16	10	98.61	5-30

Table 6-6:Summary of Best AUC

Classifier	FS Method	Colon Dataset		Leukemia Dataset	
		Best AUC	# of Features	Best AUC	# of Features
KNN	mRCD	0.84	30	0.98	200
	mRCQ	0.87	50	0.98	10,50,200
	mRDD	0.97	5	1.00	5,30
	mRDQ	0.94	20	1.00	10-100
	MRC	0.87	100	0.96	20
	MRD	0.93	20	1.00	5
	QPFS	0.82	20	0.87	50
	PLS	1.00	5	1.00	5
SVM	mRCD	0.90	50	1.00	50-200
	mRCQ	0.92	30	1.00	50
	mRDD	0.95	10	1.00	ALL
	mRDQ	0.96	100	1.00	10-200
	MRC	0.93	200	1.00	50-200
	MRD	0.93	10	1.00	50
	QPFS	0.95	5	0.99	50
	PLS	1.00	ALL	1.00	ALL
NN	mRCD	0.91	50	1.00	30
	mRCQ	0.93	10	1.00	30
	mRDD	0.94	10	1.00	ALL
	mRDQ	0.96	30	1.00	10-200
	MRC	0.92	200	1.00	20,30
	MRD	0.92	20	1.00	10,30,50
	QPFS	0.94	5	1.00	50
	PLS	1.00	5-30	1.00	5-30
RF	mRCD	0.93	50	1.00	50-200
	mRCQ	0.94	100	1.00	200
	mRDD	1.00	5	1.00	All
	mRDQ	0.99	10,50	1.00	10-200
	MRC	0.94	5	1.00	200
	MRD	0.95	30	1.00	10-100
	QPFS	0.94	5	0.98	200
	PLS	0.99	5	1.00	10-50

TABLE 6-5 and TABLE 6-6 summarize the results. TABLE 6-5 shows the best accuracy of the four classifiers with the feature selection methods for the colon and leukemia datasets. TABLE

6-6 shows the best AUC of the four classifiers with the feature selections methods for the colon and leukemia datasets. We observe the following from TABLE 6-5.

1. KNN with PLS gives 100% accuracy for colon dataset when 5 features are selected.
2. KNN with mRDD gives 100% accuracy for leukemia dataset when 5 or 30 features are selected and mRDQ give 100% accuracy when 20 – 100 features are selected.
3. SVM with PLS gives 100% accuracy for colon dataset irrespective of the number of features selected.
4. SVM with mRDD give 100% accuracy for leukemia dataset when 5- 50 features are selected; mRDQ gives 100% accuracy when 10 – 100 features are selected; MRD gives 100% accuracy when 50 features are selected; and PLS gives 100% accuracy irrespective of the number of features selected.
5. Neural Network with PLS gives 100% accuracy for colon dataset when 5 – 30 features are selected.
6. Neural Network with mRDD gives 100% accuracy for leukemia dataset when 10 – 200 features are selected; mRDQ gives 100% accuracy when 20 – 200 features are selected; and PLS gives 100% accuracy when 5 – 30 features are selected.
7. Random Forest with PLS gives the highest accuracy of 95.16% for colon dataset when 10 features are selected.
8. Random Forest with mRDD give 100% accuracy for leukemia dataset irrespective of the number of features selected; mRDQ gives 100% accuracy when 20 – 200 features are selected; MRD gives 100% accuracy when 10 features or 30 – 100 features are selected.

The following results are observed from TABLE 6-6.

1. KNN with PLS gives AUC value of 1.0 for the colon dataset when 5 features are selected.
2. KNN with mRDD gives AUC value of 1.0 for the leukemia dataset when 5 or 30 features are selected; mRDQ gives AUC value 1.0 when 10 – 100 features are selected; MRD and PLS give AUC value of 1.0 when 5 features are selected.
3. SVM with PLS gives AUC value 1.0 for colon dataset irrespective of the number of features selected.
4. SVM with all except QPFS (0.99) feature selection method give AUC value 1.0 for leukemia dataset for different number of features.
5. Neural Network with PLS give AUC value 1.0 for colon dataset when 5 – 30 features are selected.
6. Neural Network gives AUV value of 1.0 for all feature selection methods for the leukemia dataset.
7. Random Forest with mRDD give AUC value 1.0 for colon dataset when 5 features are selected.
8. Random Forest gives AUC value of 1.0 with all feature selection methods except PQFS (0.98) for the leukemia dataset.

6.4 Conclusion

In this chapter, we compared 8 different ways of feature selection preprocess methods from 4 different feature selection methods. The experiment shows that discretize can somehow improve performance of microarray data analysis. Out of eight feature selection methods, Peng's mRMR has shown considerably good performance while PLS has achieved some competitive results when 5 features were selected. From Colon dataset and Leukemia dataset, we concluded some general results above in section 6.3, which can be used for other data set.

In [8], one of the most important feature for QPFS is the computational complexity. We didn't cover the computational time in this paper. That may be a future work.

We didn't apply the discretize method onto QPFS and PLS, this could also be a future work.

Chapter 7

Conclusions and Future Work

The main target of our experiment and this study is to determine which method (Feature Selection or Feature Extraction) give better performance in accuracy or AUC when we use SVM, RF, KNN or NN to classification.

Thus, we compared the performance among these four classifiers by 10-folds cross validation and ratio comparison with Principle Component Analysis. We also compared the performance by 10-folds cross validation with 8 different methods of feature selection.

The result shows that both feature extraction method and feature selection method can somehow improve the classification results. Some suggestions rules are given in Chapter 6 for selection numbers of attributes.

To conclude, the results in this study suggested that SVM has really good and stable performance by compare to KNN, RF and NN. PCA can give improve to classification tasks when using 10-folds cross validation. For feature selection methods, some certain combinations can greatly improve the performance.

However, there are some other aspects that we didn't cover due to lack of time. For the four classifiers, we didn't measure the computational complexity as well as the computational time consumptions. Few works we mentioned in our study (QPFS, PCA, PLS) can decrease the computing time. The improvement of computing speed allows us to discover more and do further

experiments for different parameters of certain classifiers. For instance, the selection of parameter k in KNN is arguable. Some studies [49] suggested to choose 3 and some suggested to 1. More experiments can give us further suggestion of the selection.

For feature selection methods, we didn't discuss the comparison continuously. We chose the number of features fixed (5, 10, 20, 30, 50, 100, 200). The algorithm gives a rank for vary attributes and we choose the attributes by selecting the top rankings. To compare and selection the top rankings continuously could also be a possible future work to discover the relationship and variation of features.

References

[1] D. Jawdat, "The Era of Bioinformatics," in *2006 2nd International Conference on Information & Communication Technologies*, 2006, pp. 1860-1865.

[2] K.-C. Wong, *Computational Biology and Bioinformatics: Gene Regulation*: CRC Press, 2016.

[3] A. Brazma, A. Robinson, G. Cameron, and M. Ashburner, "One-stop shop for microarray data," *Nature*, vol. 403, pp. 699-700, 2000.

[4] A. L. Tarca, R. Romero, and S. Draghici, "Analysis of microarray experiments of gene expression profiling," *American journal of obstetrics and gynecology*, vol. 195, pp. 373-388, 2006.

[5] S. Choudhuri, "Microarrays in biology and medicine," *Journal of biochemical and molecular toxicology*, vol. 18, pp. 171-179, 2004.

[6] L. O. Hall, "Finding the right genes for disease and prognosis prediction," in *2010 International Conference on System Science and Engineering*, 2010, pp. 1-2.

[7] E. Byvatov and G. Schneider, "Support vector machine applications in bioinformatics," *Applied bioinformatics*, vol. 2, pp. 67-77, 2002.

[8] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, pp. 389-422, 2002.

[9] Z. R. Yang, "Biological applications of support vector machines," *Briefings in bioinformatics*, vol. 5, pp. 328-338, 2004.

[10] J. Herrero, A. Valencia, and J. Dopazo, "A hierarchical unsupervised growing neural network for clustering gene expression patterns," *Bioinformatics*, vol. 17, pp. 126-136, 2001.

[11] E. Capriotti, P. Fariselli, and R. Casadio, "A neural-network-based method for predicting protein stability changes upon single point mutations," *Bioinformatics*, vol. 20, pp. i63-i68, 2004.

[12] R. Díaz-Uriarte and S. A. De Andres, "Gene selection and classification of microarray data using random forest," *BMC bioinformatics*, vol. 7, p. 1, 2006.

[13] A. L. Boulesteix, S. Janitza, J. Kruppa, and I. R. König, "Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, pp. 493-507, 2012.

[14] P. Horton, K.-J. Park, T. Obayashi, N. Fujita, H. Harada, C. Adams-Collier, *et al.*, "WoLF PSORT: protein localization predictor," *Nucleic acids research,* vol. 35, pp. W585-W587, 2007.

[15] M.-L. Zhang and Z.-H. Zhou, "A k-nearest neighbor based algorithm for multi-label classification," in *2005 IEEE international conference on granular computing*, 2005, pp. 718-721.

[16] H.-B. Shen and K.-C. Chou, "Ensemble classifier for protein fold pattern recognition," *Bioinformatics,* vol. 22, pp. 1717-1722, 2006.

[17] L. Li, C. R. Weinberg, T. A. Darden, and L. G. Pedersen, "Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method," *Bioinformatics,* vol. 17, pp. 1131-1142, 2001.

[18] C. Ambroise and G. J. McLachlan, "Selection bias in gene extraction on the basis of microarray gene-expression data," *Proceedings of the National Academy of Sciences,* vol. 99, pp. 6562-6566, May 14, 2002 2002.

[19] M. Garc, T. a, G. F, V. mez, D. Becerra-Alonso, B. Meli, *et al.*, "Feature Grouping and Selection on High-Dimensional Microarray Data," in *2015 International Workshop on Data Mining with Industrial Applications (DMIA)*, 2015, pp. 30-37.

[20] S. Kar, K. D. Sharma, and M. Maitra, "A comparative study on gene ranking and classification methods using microarray gene expression profiles," in *Michael Faraday IET International Summit 2015*, 2015, pp. 596-600.

[21] A. Rouhi and H. Nezamabadi-pour, "A hybrid method for dimensionality reduction in microarray data based on advanced binary ant colony algorithm," in *2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, 2016, pp. 70-75.

[22] E. Pashaei, M. Ozen, and N. Aydin, "Gene selection and classification approach for microarray data based on Random Forest Ranking and BBHA," in *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, 2016, pp. 308-311.

[23] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of bioinformatics and computational biology,* vol. 3, pp. 185-205, 2005.

[24] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence,* vol. 27, pp. 1226-1238, 2005.

[25] K. Y. Yeung and W. L. Ruzzo, "Principal component analysis for clustering gene expression data," *Bioinformatics,* vol. 17, pp. 763-774, 2001.

[26] A. Wang and E. A. Gehan, "Gene selection for microarray data analysis using principal component analysis," *Statistics in medicine,* vol. 24, pp. 2069-2087, 2005.

[27] M. Jalali-Heravi and A. Kyani, "Application of genetic algorithm-kernel partial least square as a novel nonlinear feature selection method: activity of carbonic anhydrase II inhibitors," *European journal of medicinal chemistry,* vol. 42, pp. 649-659, 2007.

[28] I. Rodriguez-Lujan, R. Huerta, C. Elkan, and C. S. Cruz, "Quadratic programming feature selection," *Journal of Machine Learning Research,* vol. 11, pp. 1491-1516, 2010.

[29] Y. Prasad, K. Biswas, and P. Singla, "Scaling-Up Quadratic Programming Feature Selection," in *AAAI (Late-Breaking Developments)*, 2013.

[30] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, *et al.*, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proceedings of the National Academy of Sciences,* vol. 96, pp. 6745-6750, 1999.

[31] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, *et al.*, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *science,* vol. 286, pp. 531-537, 1999.

[32] E. D. Feigelson and G. Babu, "Statistical challenges in modern astronomy," *arXiv preprint astro-ph/0401404*, 2004.

[33] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory,* 1992, pp. 144-152.

[34] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning,* vol. 20, pp. 273-297, 1995.

[35] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning* vol. 1: Springer series in statistics Springer, Berlin, 2001.

[36] OpenCV.Org. (2016). *Introduction to SVM*. Available: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[37] C. Jin and L. Wang, "Dimensionality dependent PAC-Bayes margin bound," in *Advances in Neural Information Processing Systems*, 2012, pp. 1034-1042.

[38] T. K. Ho, "Random decision forests," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, 1995, pp. 278-282.

[39] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE transactions on pattern analysis and machine intelligence,* vol. 20, pp. 832-844, 1998.

[40] E. M. Kleinberg, "On the algorithmic implementation of stochastic discrimination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 473-490, 2000.

[41] E. Kleinberg, "An overtraining-resistant stochastic modeling method for pattern recognition," *The annals of statistics*, vol. 24, pp. 2319-2349, 1996.

[42] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5-32, 2001.

[43] F. Livingston, "Implementation of Breiman's random forest machine learning algorithm," *ECE591Q Machine Learning Journal Paper*, 2005.

[44] D. Benyamin. (2012). *A Gentle Introduction to Random Forests, Ensembles, and Performance Metrics in a Commercial System*. Available: https://citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics/

[45] S.-C. Wang, "Artificial Neural Network," in *Interdisciplinary Computing in Java Programming*, ed Boston, MA: Springer US, 2003, pp. 81-100.

[46] S. Sayad. (2010). *Artificial Neural Network*. Available: http://chem-eng.utoronto.ca/~datamining/dmc/artificial_neural_network.htm

[47] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, pp. 21-27, 1967.

[48] B. R. Kowalski and C. Bender, "k-Nearest Neighbor Classification Rule (pattern recognition) applied to nuclear magnetic resonance spectral interpretation," *Analytical Chemistry*, vol. 44, pp. 1405-1411, 1972.

[49] S. Sayad. (2010). *K Nearest Neighbors - Classification*. Available: http://www.saedsayad.com/k_nearest_neighbors.htm

[50] S. A. Quadri. (2012). *Feature Extraction and Principal Component Analysis*. Available: http://www.slideshare.net/reachquadri/feature-extraction-and-principal-component-analysis

[51] J. Brownlee. (2014). *An introduction to feature selection*. Available: http://machinelearningmastery.com/an-introduction-to-feature-selection/

[52] K. Peason, "On lines and planes of closest fit to systems of point in space," *Philosophical Magazine*, vol. 2, pp. 559-572, 1901.

[53] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of educational psychology*, vol. 24, p. 417, 1933.

[54] R. P. Brent, F. T. Luk, and C. Van Loan, "Computation of the generalized singular value decomposition using mesh-connected processors," in *27th Annual Techincal Symposium*, 1983, pp. 66-71.

[55] I. Jolliffe, *Principal component analysis*: Wiley Online Library, 2002.

[56] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics,* vol. 2, pp. 433-459, 2010.

[57] P. J. Shaw, *Multivariate statistics for the environmental sciences*: Wiley, 2009.

[58] V. K. Nagaraja and W. Abd-Almageed, "Feature Selection using Partial Least Squares regression and optimal experiment design," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1-8.

[59] P. Geladi and B. R. Kowalski, "Partial least-squares regression: a tutorial," *Analytica chimica acta,* vol. 185, pp. 1-17, 1986.

[60] S. Wold, M. Sjöström, and L. Eriksson, "PLS-regression: a basic tool of chemometrics," *Chemometrics and intelligent laboratory systems,* vol. 58, pp. 109-130, 2001.

[61] J. L. Lustgarten, V. Gopalakrishnan, H. Grover, and S. Visweswaran, "Improving classification performance with discretization on biomedical datasets," in *AMIA annual symposium proceedings*, 2008, p. 445.

[62] R. Abraham, J. B. Simha, and S. Iyengar, "A comparative analysis of discretization methods for medical datamining with naïve Bayesian classifier," in *Information Technology, 2006. ICIT'06. 9th International Conference on*, 2006, pp. 235-236.

[63] A. Tillander, "Effect of data discretization on the classification accuracy in a high‐dimensional framework," *International Journal of Intelligent Systems,* vol. 27, pp. 355-374, 2012.

[64] M. R. Chmielewski and J. W. Grzymala-Busse, "Global discretization of continuous attributes as preprocessing for machine learning," *International journal of approximate reasoning,* vol. 15, pp. 319-331, 1996.

[65] J. Catlett, "On changing continuous attributes into ordered discrete attributes," in *European working session on learning*, 1991, pp. 164-178.

[66] J. Cerquides and R. L. De Màntaras, "Proposal and Empirical Comparison of a Parallelizable Distance-Based Discretization Method," in *KDD*, 1997, pp. 139-142.

[67] S. W. Hadley, C. Pelizzari, and G. Chen, "Registration of localization images by maximization of mutual information," in *Proc. Ann. Meeting of the Am. Assoc. Physicists in Medicine*, 1996.

[68] N. Kwak and C.-H. Choi, "Input feature selection by mutual information based on Parzen window," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, pp. 1667-1671, 2002.

[69] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics,* vol. 33, pp. 1065-1076, 1962.

[70] H.-J. Reinhardt, *Analysis of approximation methods for differential and integral equations* vol. 57: Springer Science & Business Media, 2012.

[71] R. A. Young, "Biomedical Discovery with DNA Arrays," *Cell,* vol. 102, pp. 9-15, 7/7/ 2000.

[72] H. C. Liu, P. C. Peng, T. C. Hsieh, T. C. Yeh, C. J. Lin, C. Y. Chen, *et al.*, "Comparison of Feature Selection Methods for Cross-Laboratory Microarray Analysis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* vol. 10, pp. 593-604, 2013.

[73] A. Antoniadis, S. Lambert-Lacroix, and F. Leblanc, "Effective dimension reduction methods for tumor classification using gene expression data," *Bioinformatics,* vol. 19, pp. 563-570, 2003.

[74] J. Kittler, "Pattern recognition. A statistical approach," 1982.

[75] S. Geisser, *Predictive inference* vol. 55: CRC press, 1993.

[76] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai,* 1995, pp. 1137-1145.

[77] J. Schneider. (1997). *Cross validation*. Available: https://www.cs.cmu.edu/~schneide/tut5/node42.html

[78] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," pp. 40-79, 2010 2010.

[79] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering,* vol. 40, pp. 16-28, 1// 2014.

[80] S. De Jong, “SIMPLS: An alternative approach to partial least squares regression,” Chemometrics and Intelligent Laboratory Systems, vol. 18, no. 3, pp. 251–263, 1993

APPENDIX

Appendix A: Matrix I2000 and Names of each Gene in Colon Dataset

Matrix I2000 is available in http://genomics-pubs.princeton.edu/oncology/affydata/I2000.html

The names of each gene expression is available in:

http://genomics-pubs.princeton.edu/oncology/affydata/names.html

Below is an example for the first 4 genes.

Hsa.3004	H55933	3' UTR	1	203417	H.sapiens mRNA for homologue to yeast ribosomal protein L41.	Hsa.3004
Hsa.13491	R39465	3' UTR	2a	23933	EUKARYOTIC INITIATION FACTOR 4A (Oryctolagus cuniculus)	Hsa.13491
Hsa.13491	R39465	3' UTR	2a	23933	EUKARYOTIC INITIATION FACTOR 4A (Oryctolagus cuniculus)	Hsa.13491
Hsa.37254	R85482	3' UTR	2a	180093	SERUM RESPONSE FACTOR (Homo sapiens)	Hsa.37254
…						

Appendix B: Codes

Table 0-1:Discritization using Python

```
'''
excel_discrete.py
This program is to discrete the input csv file.
p is the mean value and q is the standard deviation

1: if the value in (p+q/2, + infinite)
0: if the value in [p-q/2,p+q/2]
-1: if the value in (-infinite, p-q/2)

'''
def readfile(filename,p,q):
        import csv

        with open(filename,'rb') as f:
                reader = csv.reader(f)
                rows = [row for row in reader]
                n_column = len(rows[0])
                n_row =len(rows)
                print rows[1][15]
                print p+q/2
                for i in range(1,n_column):
                        for j in range(1,n_row):
                                if float(rows[j][i])> (p+q/2):
                                        rows[j][i]=1.0
                                elif float(rows[j][i]) < (p-q/2):
                                        rows[j][i]=-1.0
                                else:
                                        rows[j][i]=0.0
                print rows[1][15]
                print type(rows[1][15])
        f.close()
        return rows


def write_csv(filename, list_towrite):
        import csv
        with open(filename,'w') as f:
                f_csv = csv.writer(f)
                f_csv.writerows(list_towrite)
        f.close()

inputfile = "colon_class_front.csv"
```

```
# inputfile = "example.csv"
outputfile = "c_dicreted_3t.csv"
rows=readfile(inputfile,0.0,1.0)
write_csv(outputfile, rows)
```

Table 0-2:Transform format using Python

```
# -*- coding: utf-8 -*-
'''
transform.py
Input: mRMR output files. Format:
1        3320    Attr3320        0.459
2        6281    Attr6281        0.425
3        804     Attr804         0.391
4        6184    Attr6184        0.376
5        1962    Attr1962        0.344
6        1829    Attr1829        0.344
7        2121    Attr2121        0.334
8        1674    Attr1674        0.334
9        6677    Attr6677        0.325
10       2363    Attr2363        0.324
11       4847    Attr4847        0.318
12       2439    Attr2439        0.316
13       1087    Attr1087        0.306
...

OUTPUT: Two files. Format:
File1: MaxRel Feature selector
        3320,6281,804, ...
File2: mRMR Feature selector
        3320,2363,1674, ...

This program is use for preprocess file select.txt for excel_file.py.
We use the output result grep filtered as the input file, we import the
elements we need by order and output them separated by comma.

'''

def read_txt(filename):
        contents = []
        with open(filename) as f:
                for line in f:
                        content = line.split('\t')
                        contents.extend([content])
        f.close()
```

```
        return contents

def write_txt(contents,filename_max,filename_mrmr,number):
        import csv
        maxRel="0,"
        mrmr="0,"
        for i in range(0,number):
                maxRel += contents[i][1].strip()+ ','
        maxRel=maxRel[0:len(maxRel)-1]
        for i in range(number,number<<1):
                mrmr += contents[i][1].strip()+ ','
        mrmr=mrmr[0:len(mrmr)-1]

        with open(filename_max,'w') as f:
                f.write(maxRel)
        f.close()
        with open(filename_mrmr,'w') as f:
                f.write(mrmr)
        f.close()




'''
Main
Change feature_number to the attribute you want to select.
5,10,20,30,50,100,200 in this experiment.

'''
feature_number = 200
# inputfile="example_trans.txt"
inputfile_l="mRMR_algorithm_files/leu_con_MIQ" + str(feature_number) +".txt"
inputfile_c="mRMR_algorithm_files/colon_con_MIQ" + str(feature_number)
+".txt"
MaxRel_output_l = "MaxRel_select_l.txt"
mRMR_output_l = "mRMR_select_l.txt"
MaxRel_output_c = "MaxRel_select_c.txt"
mRMR_output_c = "mRMR_select_c.txt"
contents=read_txt(inputfile_l)
write_txt(contents,MaxRel_output_l,mRMR_output_l,feature_number)
contents=read_txt(inputfile_c)
write_txt(contents,MaxRel_output_c,mRMR_output_c,feature_number)
```

Table 0-3:Subset Extraction using Python

```
# -*- coding: utf-8 -*-
'''
excel_file.py
This program is to select the sub-dataset from original dataset.
We use select file and feature_number to control input.
The variable feature_number need to be change manually.

'''

def readfile(filename):
        import csv

        with open(filename,'rb') as f:
                reader = csv.reader(f)
                rows = [row for row in reader]
                lens = len(rows[0])
                columns = []
                column = []
                for i in range(0,lens):
                        column = [col[i] for col in rows]
                        columns.extend([column])
                        # print column
                # print '\n=====\n'
        f.close()
        return columns

def read_select(filename):
        with open(filename) as f:
                for line in f:
                        nums = line.split(',')
                        nums = map(int, nums)
                        return nums

def write_csv(filename, list_towrite):
        import csv
        with open(filename,'w') as f:
                f_csv = csv.writer(f)
                f_csv.writerows(list_towrite)
        f.close()

def select(datafile,outputfile,selector):
        columns_toselect = read_select(selector)
        columns_all = readfile(datafile)
        new_set=[]
```

```
        for i in columns_toselect:
                new_set.extend([columns_all[i]])
        zipped = zip(*new_set)
        write_csv(outputfile,zipped)


'''
Main
'''
feature_number = 200
inputfile1="colon_cf_tf.csv"
inputfile2="leu_cf_tf.csv"
outputfile1="datafiles/MaxRel_tf/colon_con_max_" + str(feature_number)
+".csv"
outputfile2="datafiles/mRMR_MIQ/colon_c_Q_" + str(feature_number) +".csv"
outputfile3="datafiles/MaxRel_tf/leu_con_max_" + str(feature_number) +".csv"
outputfile4="datafiles/mRMR_MIQ/leu_c_Q_" + str(feature_number) +".csv"
selectfile1="MaxRel_select_c.txt"
selectfile2="mRMR_select_c.txt"
selectfile3="MaxRel_select_l.txt"
selectfile4="mRMR_select_l.txt"
# print read_select("MaxRel_select.txt")
select(inputfile1,outputfile1,selectfile1)
select(inputfile1,outputfile2,selectfile2)
select(inputfile2,outputfile3,selectfile3)
select(inputfile2,outputfile4,selectfile4)
```